A Secure Key Management Interface with Asymmetric Cryptography

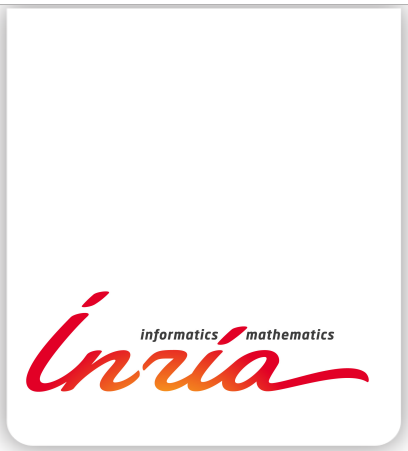
Marion Daubignard, David Lubicz, Graham Steel

A Secure Key Management Interface with Asymmetric Cryptography

Marion Daubignard, David Lubicz, Graham Steel

RESEARCH REPORT
N° 8274
April 2013
Project-Team Prosecco

ISSN 0249-6399 ISRN INRIA/RR--8274--FR+ENG



A Secure Key Management Interface with Asymmetric Cryptography

Marion Daubignard*, David Lubicz*, Graham Steel

Project-Team Prosecco

Research Report n° 8274 — April 2013 — 29 pages

* DGA.MI, BP 57419 35174 Bruz Cedex, France

RESEARCH CENTRE
PARIS – ROCQUENCOURT

Domaine de Voluceau, - Rocquencourt
B.P. 105 - 78153 Le Chesnay Cedex

Abstract: Cryptographic devices such as Hardware Security Modules are only as secure as their application programme interfaces (APIs) that offer cryptographic functionality to the outside world. Design flaws and implementation errors in security APIs have been shown to cause vulnerabilities that may leak secrets such as keys and PINs. Ideally, we would like to design such interfaces in such a way that we can formally prove security properties, even in the presence of some corrupted keys. In this work, we take such a design for a provably secure interface for symmetric key management, due to Cortier and Steel, and extend it to asymmetric cryptography, giving new security definitions and associated proofs. Asymmetric cryptography forces us to consider confidentiality and integrity properties separately and provide support for classical operations of public key infrastructure (e.g. certification of public keys). As far as we are aware this is the first such provably secure interface to support asymmetric key operations for key management: Cachin and Chandran's secure token interface supports asymmetric key operations only for encrypting and signing data, not for managing keys.

Key-words: crytography, key management, security APIs

Une généralisation de l'API Cortier-Steel pour la cryptographie asymétrique

Résumé : Les systèmes cryptographiques tels que les modules matériels de sécurité ne peuvent apporter de garanties de sécurité que dans la mesure où leur interface de programmation (API), qui offre les services de cryptographie à l'extérieur de module, atteint un certain niveau de sécurité. Il a été constaté que des défauts de conception ou des erreurs d'implémentation dans les APIs de sécurité sont à l'origine de vulnérabilités pouvant entraîner la fuite de secrets comme des clefs ou des PINs. Idéalement, nous voudrions concevoir de telles interfaces de manière à pouvoir prouver formellement des propriétés de sécurité, même si certaines clefs sont corrompues. Dans cet article, nous partons d'une telle API, due à Cortier et Steel, conçue de manière à disposer d'une preuve de sécurité pour la gestion de clefs symétriques, et nous l'adaptons à la cryptographie asymétrique en donnant une nouvelle définition de sécurité avec les preuves associées. Afin de prendre en compte la cryptographie asymétrique, nous sommes amenés à gérer de manière différentiée les propriétés de confidentialité et d'intégrité et à ajouter les fonctionnalités classiques d'une infrastructure de gestion de clefs publiques (i.e. la certification des clefs publiques). À notre connaissance, il s'agit de la première preuve d'interface prouvée permettant l'usage de primitives asymétriques pour la gestion de clefs : l'interface de Cachin et Chandran prévoit l'usage de primitives asymétriques uniquement pour le chiffrement et la signature de données, et non pas pour la gestion des clefs.

Mots-clés : cryptographie, gestion des clés, sécurité des APIs

1 Introduction

In a context of constant security threats combined with increasing heterogeneity of platforms and applications, developers are turning more and more to solutions based on secure hardware, whether it be a smartcard, Trusted Platform Module (TPM), hardware-secured virtual execution environment (e.g. TrustZone) or Hardware Security Module (HSM). In a typical architecture, the secure hardware contains cryptographic keys and the ability to perform some basic crypto operations which can be leveraged to ensure security for the whole system. However, designing the application programme interface (API) of such a device is difficult: it must allow the user to manage the keys on the device and access the crypto without allowing an attacker, who may in the worst case be able to make arbitrary calls to the API, to be able to obtain secrets. Many attacks have been found on the APIs of contemporary devices [2, 3, 5]. One promising approach to solving this problem is to design APIs such that one can formally prove security properties in the presence of a suitably powerful intruder. Such an approach has been applied both in the standard cryptographic model [4] and the symbolic or Dolev-Yao model [7].

However, neither of these designs present a scheme for managing keys using asymmetric cryptography, which is widely used in practice for the task since it provides a convenient way to bootstrap security without any pre-shared secrets. The contribution of this paper is to present the design for such an API with security proofs in the symbolic model. For the symmetric key part of the API, we adapt slightly the API designed by Cortier and Steel [7]. For the asymmetric key part, we have to solve a problem that doesn't arise in the symmetric key case: so-called "Trojan key" attacks [6]. Since anyone can encrypt under a public key, we have to add an explicit mechanism for assuring the integrity of keys to be imported onto a device. We add signature keys for signing encryption under public keys and also separate certification keys, the latter used to manage the public key infrastructure (PKI) of keys and certificates. We show how to adapt the security labels given to keys by Cortier and Steel to this new scenario, with separate labels for confidentiality of the private key and integrity of the corresponding public key. This allows us to account for corruption in our proof. As far as we are aware, this is the first such design to be proposed with security proofs.

This report is organized as follows. We start with an introduction of our symbolic model and explain the features of our API design in Section 2. We describe the API rules formally in Section 3, and then give the security properties and sketch their proof in Section 4. Then, the complete details of the proof can be found in Section 5. We describe some experiments implementing protocols with the API in Section 6 and draw conclusions in Section 7.

Related Work Cortier and Steel (CS) [7] proposed an API that supports only symmetric key cryptography, but can nonetheless be used to implement any secure symmetric key exchange protocol from the Clark-Jacob corpus. The main principle is that keys are arranged in a hierarchy of levels. Each key is associated to its level and the set of agents who are allowed to use it. This association is made when storing the key on the device, by including it as metadata stored with the key, and when encrypting the key for transfer, by tagging the encrypted key with exactly this information. The API rules are designed such that keys may only be encrypted by other keys which are higher in the hierarchy, i.e. they are at least one level higher and assigned to a set of agents that is equal to or smaller than the payload key. We generalise this notion slightly in our API. The CS API includes a notion of freshness for imported keys enforced by nonces. It has also recently been extended to accommodate key revocation [8]. Although we do not include these mechanisms in our API, we do not foresee any obstacle to these generalizations if needed.

Cachin and Chandran proposed an API with a quite different design [4]. They rely on the fact that all keys are stored on a central key server. Instead of assigning security attributes such as levels and agent identifiers to keys at creation time, they allow the key's role to evolve over time by logging all operations, and then disallowing operations that would be insecure by observing the log. They allow

asymmetric keys to be managed by symmetric key cryptography, but do not allow asymmetric keys to be used for key management operations like export and import.

Other work has investigated the foundations of models for secure key management APIs: Kremer, Steel and Warinschi give a model that can be interpreted in the symbolic and computational cryptography worlds [11]. They show that the possibility of key corruption requires strong assumptions to be made on the key wrapping primitives in the computational model. Recent work by Künnemann, Kremer and Steel investigates composable notions of security for key management [10]. This is an appealing idea because it allows (almost) arbitrary secure cryptographic primitives to be used with the keys under management without having to repeat the security proofs, but currently only management with symmetric keys is supported.

2 Design of the API

We present the design of our API in an abstract 'Dolev-Yao' style symbolic model. We first describe the roles assigned to keys in our API. We then give the syntax and informal semantics for the message algebra and introduce our notion of *key handles* which extends previous designs.

2.1 Key Types

In order to limit the number of key roles in the API we consider that the asymmetric keys are double keys, with one part for encryption/decryption, and one for signature/verification. This means that the same key can be used as an input of both an encryption and a signature scheme. Thus, we have encryption/verification public keys and decryption/signature private keys. It is clear that in practise a double key can simply be obtained by the concatenation of a signature and encryption key and that a simple key can be simulated by a double key. Thus, we do not lose generality with this simplification. Moreover, it makes sense from a security point of view since the encrypt and sign operation is the minimal basic operation which ensures the confidentiality of message and an authentication of the issuer, which is mandatory for the set up of our security policy. Signature keys are used to sign encryptions of other keys or messages. Asymmetric public keys are certified by certification keys (with a signature algorithm).

The list of key roles that we are going to manipulate is:

- symmetric encryption/decryption keys;
- encryption/verification of signature double public keys;
- decryption/signature double private keys;
- verification of certificates public keys;
- certification private keys.

It is possible that the algorithm used to sign the certificates is the same as the one used to sign the encrypted messages. Nonetheless, it is important to distinguish the key roles to prevent a signature algorithm from being used as a certification oracle by an adversary. The different key roles and their associated types are summarised in the table 1. $\mathcal{T}$ denotes the set of key types.

2.2 Security Levels

The set of key security levels I is a finite set together with an partial strict order relation denoted $<$. We suppose that there is a minimal element in I denoted by 0. By definition, for all $x \in I \setminus 0$, we have $0 < x$. The 0 element represents the security level of public information. We are given a partition of I in two subsets:

Key	Role	Type
Priv	decryption/signature private key	privDecSign
Pub	encryption/verification of signature public key	pubEncVerif
Sym	symmetric encryption key	symEncDec
pub	certificate verification key	pubCertVerif
priv	certificate signature key	privCertSign

Figure 1: Table of the set of key roles and types ($\mathcal{T}$)

- the levels $I_1 \subset I$ which correspond to the keys which can only deal with regular messages;
- the levels $I_2 \subset I$ which correspond to the keys which can be used to transport keys of level I_1.

We set $I_{>0} = I_1 \cup I_2 = I - \{0\}$.

2.3 Message Algebra

Messages are represented by a term algebra. We suppose given a set of agents Agent, a set of nonces Nonce and a set of keys Key. We are also given a set of variables Var in which we distinguish a set of key variables VarKey and a set of nonce variables VarNonce. All these sets are countably infinite. The term algebra is given by:

$$\begin{array}{rcl} \mathsf{Keyv} & ::= & \mathsf{Key} \mid \mathsf{VarKey} \mid inv(\mathsf{Keyv}) \\ \mathsf{Noncev} & ::= & \mathsf{Nonce} \mid \mathsf{VarNonce} \\ \mathsf{Msg} & ::= & \mathsf{Agent} \mid \mathsf{Keyv} \mid \mathsf{Noncev} \mid I \mid \mathcal{T} \mid \{\!|\mathsf{Msg}|\!\}_{\mathsf{Keyv}} \mid \{\mathsf{Msg}\}_{\mathsf{Keyv}} \\ & & \mid \Sigma(\mathsf{Msg}, \mathsf{Keyv}) \mid \mathsf{nhdl}(\mathsf{Msg}) \mid < \mathsf{Msg}, \mathsf{Msg} > \\ \mathsf{Handle} & ::= & h^{\alpha}_{\mathsf{Agent}}(\mathsf{Noncev}, \mathsf{Noncev}, \mathsf{Msg}, \mathcal{T}, I, \mathscr{S}, \mathscr{S}) \\ & & \mid h_{\mathsf{Agent}}(\mathsf{Noncev}, \mathsf{Msg}) \end{array}$$

where $\mathscr{S}$ is the set of subsets of Agent.

The set Keyv represents the set of keys and variable of keys. A term of the form $inv(k)$ with $k \in \mathsf{Key}$ represents the private key associated to the public key k. The set Noncev is the set of nonces and variable of nonces. The terms of type Msg are made of elements of Agent, Keyv, Noncev together with constructors representing encryption, signature together with sets needed to represent the attributes of the handles. More precisely,

- the term $\{\!|m|\!\}_k$ represents the symmetric encryption of the message m with the key k;
- the term $\{m\}_k$ represents the asymmetric encryption of the message m with the double key k;
- the term $\Sigma(m, k)$ represents the signature of the message m with the double key k;
- the term nhdl() allows one to encapsulate a regular message which does not correspond to the transportation of a handle (see below);
- the term $< m_1, m_2 >$ represents the pair of the two messages $m_1, m_2 \in \mathsf{Msg}$.

For $n > 0$, $< m_1, < m_2, < \ldots, m_n >>>$ is shortened as $m_1, \ldots, m_n$.

2.4 Handles

The purpose of a key management API is to give access to cryptographic functionalities without giving direct access to sensitive keys stored on the device. Instead, an agent can manipulate the data by calling the API commands and referring to the keys by their *handles*. A key handle can be thought of as a name for a key, or as a pointer to that key. Knowing the value of a key handle does not give any information about the cryptographic value of the key. We have two types of handles:

- *key handles* used to protect integrity and confidentiality of the data on the device. They are typically used for keys and secret nonces.
- *integrity handles* used to protect the integrity of data on the device. They are typically used for certificates that have been verified.

Key handles are terms of the form $h_a^{\alpha}(N_1, N_2, m, T, i, S_1, S_2)$, with:

- the agent $a \in \mathsf{Agent}$ who owns the handle;
- the identifier $N_1 \in \mathsf{Nonce}$ (unique in the whole system) of the handle;
- if m is a double private key, then N_2 is the identifier of the associated certificate of the double public key, else $N_2 = \mathsf{Null}$;
- the message $m \in \mathsf{Msg}$ (usually m is a key or a nonce) associated to the handle;
- the type $T \in \mathcal{T}$ of the message (see table 1 for a list of possible types);
- the triple $(i, S_1, S_2) \in I \times \mathscr{S} \times \mathscr{S}$ is the security level of the handle (the security policy of the API is based on this structure);
- the label $\alpha \in \{r, g\}$ allows to distinguish the keys which have been generated by a ($\alpha = g$) from the keys which have been received and imported ($\alpha = r$).

Integrity handles are terms of the form $h_a(N_1, m)$ with an identifier N_1 and a message $m \in \mathsf{Msg}$. They are meant to model the preservation of the integrity of data by a signature : given as input a valid signature of a message m, the API produces an integrity handle containing the message m. Contrarily to the common usage in cryptography where a public key certificate corresponds to signed public information, we distinguish here two elements, the pre-certificate and the certificate which is a signed pre-certificate. Indeed, the outcome of the certificate verification operation is a new pre-certificate stored under an integrity handle in the device.

In the following, for clarity, we use the notation $\mathcal{C}(N_1, N_2, N_3, k, T, i, S_1, S_2)$, which is a synonym of the concatenation of the terms $N_1, N_2, N_3, k, T, i, S_1, S_2 \in \mathsf{Msg}$, to represent a certificate of double public key. We emphasize that the notation $\mathcal{C}(N_1, N_2, N_3, k, T, i, S_1, S_2)$ does not imply requirements on the type of the fields. Nonetheless, we say that a pre-certificate is *well-formed* if its fields correspond to the following terms and types (we also give their semantic):

- the identifier $N_1 \in \mathsf{Nonce}$ of the certificate;
- the identifier $N_2 \in \mathsf{Nonce}$ of the associated private key;
- the identifier $N_3 \in \mathsf{Nonce}$ of the certification public key which allows to verify the certificate;
- a double public key $k \in \mathsf{Key}$;
- the type $T \in \mathcal{T}$ of k ;
- the associated private key handle security level $(i, S_1, S_2) \in I \times \mathscr{S} \times \mathscr{S}$.

Thus, we typically store a pair of matching asymmetric double keys as:

- a key handle $h^{\alpha}_{\cdot}(N_1, N_2, k, \text{privDecSign}, i, S_1, S_2)$ for the secret part,
- an integrity handle $h_{\cdot}(N_2, \mathcal{C}(N_2, N_1, N_3, k, \text{pubEncVerif}, i, S_1, S_2))$ for the certificate of the public part.

We remark that we choose to trace the association of public and private part of asymmetric key pairs via their identifiers. This requires to have system-wide identifiers for handles, in the sense that identifiers are independent of the secure hardware they are stored in. As a result, when importing a pre-certificate in a device, the identifier cannot be generated at random. This explains why the pre-certificate contains a field corresponding to its identifier.

2.5 API Rules

The model that we present is a transition system inspired by [7]. The state of the system is given by the family $\{\mathcal{S}_b | b \in \mathsf{Agent} \cup \{\mathsf{int}\}\}$. We also consider a set of predicates $\mathcal{P} = \{P_a | a \in \mathsf{Agent} \cup \{\mathsf{int}\}\}$, where int is a particular element representing the knowledge of the attacker. The notations $P_b(t)$ and $t \in \mathcal{S}_b$ are equivalent.

The API is represented by a set of rules of the general form:

$$P_1(u_1), \ldots, P_k(u_k) \overset{N_1,\ldots,N_m}{\Longrightarrow} P_{k+1}(v_{k+1}), \ldots, P_l(v_l),$$

where the u_i, v_i are terms, the N_i are variables and the P_i are predicates. These rules are instantiated by substituting the variables by terms of the same type. In order to explain that, let $x_1, \ldots, x_n$ be elements of Var and let $t_1, \ldots, t_n$ be a set of terms. We denote by $\{x_1 \to t_1, \ldots, x_n \to t_n\}$ the substitution σ which replaces the variables x_i by the terms t_i for $i = 1, \ldots, n$. We say that σ is well-typed if the variables x_i and the terms t_i have the same types. In the following, we only consider well-typed substitutions. The application of the substitution σ on the term t is denoted by $t\sigma$. Classically, given a set of rules, we say that a state $\mathcal{S}'$ is reachable from a state $\mathcal{S}$ if there exists an instantiated rule in the set allowing to transition from $\mathcal{S}$ to $\mathcal{S}'$. We then generalize this reachability definition to the transitive closure of a set of rules, which we denote $\Rightarrow^*$.

2.6 A model of corruption

In stating our security properties, we assume that some keys stored on secure hardwares might be lost, perhaps due to side channel attacks or other out-of-model events. We model this as shown in Figure 2: we assume that an attacker can send arbitrary commands to all devices, and additionally, has access to all the keys stored on those devices we refer to as belonging to *dishonest* agents. Other devices are referred to as *honest*. This corruption model defines an order relation on the set of keys. To a key k we can associate the set S_k of devices the corruption of which implies that of a key. A key k_1 is more secure than k_2 if $S_{k_1} \subseteq S_{k_2}$. In other words, a key is all the more secure that there are less choices in the devices the adversary can corrupt in order to compromise it.

3 Symbolic Security of the API

3.1 Security ordering

In the rules of our API, we put to use an order relation on the set of triples $(i, S_1, S_2) \in I \times \mathscr{S} \times \mathscr{S}$ (recall that $\mathscr{S}$ is the set of subsets of Agent). Let $(i_1, S_{1,1}, S_{1,2})$ and $(i_2, S_{2,1}, S_{2,2})$ be two elements of $I \times \mathscr{S} \times \mathscr{S}$, we write

$$(i_1, S_{1,1}, S_{1,2}) \prec (i_2, S_{2,1}, S_{2,2}) \text{ if } i_1 < i_2,\ S_{2,1} \subseteq S_{1,1} \text{ and } S_{2,2} \subseteq S_{1,2},$$
$$(i_1, S_{1,1}, S_{1,2}) \preceq (i_2, S_{2,1}, S_{2,2}) \text{ if } i_1 \leq i_2,\ S_{2,1} \subseteq S_{1,1} \text{ and } S_{2,2} \subseteq S_{1,2}.$$

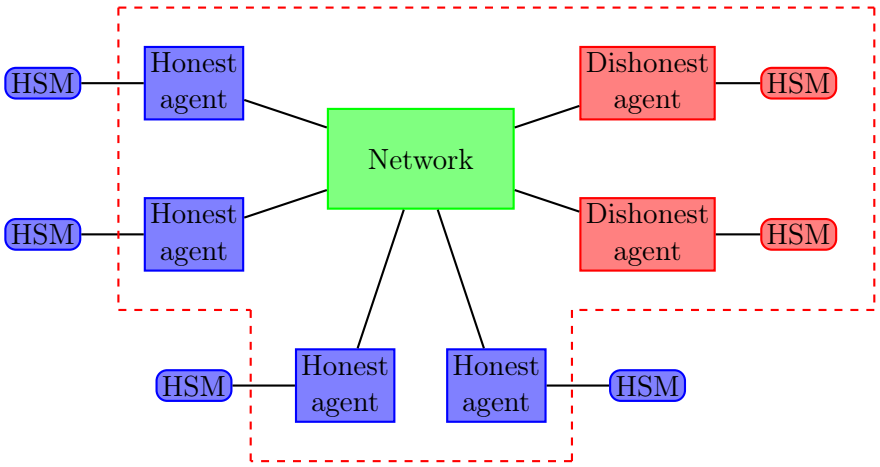


Figure 2: Corruption model

It is clear that $\preceq$ (resp. $\prec$) is an order relation (resp. a strict order relation). This relation plays an important role in the definition of the security policy of our API and the fact that it is strict ensures that we avoid cycles of encryption, e.g. terms of the form $\{\!|\{\!|\{\!|\ldots|\!\}_{K_1}|\!\}_{K_2}|\!\}_{K_1}$.

This order relation may look complex but is in fact quite natural. The security level of a handle is given by a set of devices S such that the corruption of any member of $S = (S_1, S_2)$ would imply the corruption of the handle. In the API, we want to guarantee that if a particular set $S = (S_1, S_2)$ of agents are honest (i.e. have uncorrupted HSMs), then a handle cannot be corrupted. In the case of a public key API, the keys are split into a public part (the certificate), whose value is known to everyone but the integrity of which must be guaranteed, and the private part which must be protected in confidentiality and integrity. The security of a key depends on both parts, but still it is important to be able to distinguish between these two aspects of security because we want to control the diffusion of the private key, while the integrity of the public part may depend on a long chain of certification.

In other words, if (i, S_1, S_2) is a handle level, S_1 should represent a list of agents upon whom the integrity of the public part of the key depends, i.e., if any of these agents are corrupted, then the integrity of the key is lost. The set S_2 gives the legitimate users of the private key. If any of these users are corrupted, then the private key is no longer confidential amongst these users. If the handle contains a symmetric key then S_2 has exactly the same meaning as the set S in the Cortier-Steel API.

For asymmetric keys, it may well be the case that S_1 is a rather large set (e.g. tracing a certification chain back to a root certificate) and yet we still want S_2 to be as small as possible (possibly just the user who generated the key). Finally, it should be remarked that a key k which is wrapped by another asymmetric key k' should inherit from k' the control sets S_1 and S_2 even if k is symmetric.

Dividing the agent sets into public key and private key parts also affects our security properties. In the Cortier-Steel API, a secret key cannot be sent to an agent $a \in \mathsf{Agent}$ outside of the control set S: indeed, it would be a violation of the security property in the case that a is a corrupt agent. In our setting, the security property guarantees the secrecy of a private key k if none of the agents of $S_1 \cup S_2$ are corrupted. We also want to ensure that no agent in $S_1 - S_2$ actually obtains the value of k, which they should not since they are not legitimate users of the key. Both these security requirements appear in the statement of the main result of this paper (see Theorem 1). Identifying rightful users constitutes another important motivation for dividing the control set into two parts.

3.2 The rules of the generic asymmetric API

We describe the transition rules defining the security API. We recall that agents do not know keys and use handles to refer to them. Consequently, if an agent wants to export a key, he provides its handle as an input to the corresponding API function, which replaces this latter by the value of the key and its attributes when computing the real payload value to encrypt. Reciprocally, the injection functions must identify these patterns and create the appropriate handle rather than output the key value as a plaintext. Thus, we emphasize that there has to exist a distinction between handle translations and regular messages, which we materialize by the message container nhdl. Respect of the security ordering is enforced by appropriate checks when encrypting and decrypting payloads.

In the following rules, $N_i \in \mathsf{Noncev}$, $X_k, inv(X_k), Y_k, inv(Y_k) \in \mathsf{Keyv}$, $S_i \subseteq \mathsf{Agent}$ and i (possibly indexed by an agent name) denotes an element in I.

Symmetric key generation This rule allows the generation of key X_k of level i and control sets (S_1, S_2) by the agent e for the set of users S_2, which is modeled by the following handle creation:

$$i, S_1, S_2 \overset{N,X_k}{\Longrightarrow} P_e(h^g_e(N, \mathsf{Null}, X_k, \text{symEncDec}, i, S_1, S_2 \cup \{e\})) \qquad \textbf{(Sym Gen)}$$

Symmetric encryption This rule allows agent b to encrypt with the key X_k (to which he has a handle), a payload consisting of messages and handles $m_1, \ldots, m_n$, where handles are translated into key values and attributes.

$$\begin{aligned} &P_b(h^\alpha_b(N, \mathsf{Null}, X_k, \text{symEncDec}, i, S_1, S_2)), P_b(m_1), \ldots, P_b(m_n) \\ \Longrightarrow &P_b(\{\!|m'_1, \ldots, m'_n|\!\}_{X_k}), \end{aligned} \qquad \textbf{(Sym Encrypt)}$$

with $b \in S_2$, $m_j, m'_j \in \mathsf{Msg}$ and for $j = 1, \ldots, n$:

- if $m_j = h^\alpha_b(N_j, N'_j, X_{k,j}, T_j, i_j, S_{j,1}, S_{j,2})$ with $X_{k,j} = \mathsf{Keyv} \cup \mathsf{Noncev}$ then
  - if $i \in I_2$, $b \in \mathsf{Agent}$ and $(i_j, S_{j,1}, S_{j,2}) \prec (i, S_1, S_2)$ then we let $m'_j = N_j, N'_j, X_{k,j}, T_j, i_j, S_{j,1}, S_{j,2}$;
  - else $m'_j = \varnothing$.
- else $m'_j = \mathsf{nhdl}(m_j)$.

Symmetric decryption The following rule lets agent b, provided he knows a handle pointing to key X_k, decrypt a ciphertext. Whenever a pattern consisting of a key and attributes is identified, it results in a suitable handle creation. Otherwise, the plaintext is output.

$$\begin{aligned} &P_b(h^\alpha_b(N, \mathsf{Null}, X_k, \text{symEncDec}, i, S_1, S_2)), P_b(\{\!|m_1, \ldots, m_n|\!\}_{X_k}) \\ \Longrightarrow &P_b(m'_1), \ldots, P_b(m'_n), \end{aligned} \qquad \textbf{(Sym Decrypt)}$$

with $b \in S_2$, $m_j, m'_j \in \mathsf{Msg}$ and moreover for $j = 1, \ldots, n$:

- if $m_j = N_j, N'_j, X_{k,j}, T_j, i_j, S_{j,1}, S_{j,2}$, then
  - if $i \in I_2$, $(i_j, S_{j,1}, S_{j,2}) \prec (i, S_1, S_2)$ then we set $m'_j = h^r_b(N_j, N'_j, X_{k,j}, T_j, i_j, S_{j,1}, S_{j,2})$;
  - else $m'_j = \varnothing$.
- else
  - if $m_j = \mathsf{nhdl}(t_j)$ with $t_j \in \mathsf{Msg}$ then $m'_j = t_j$;
  - else $m'_j = \varnothing$.

Asymmetric encryption/signature double key generation The following rule allows agent e, given a certification key pair under handles[1], to generate $(X_k, inv(X_k))$ of level i_2 and control sets (S_1, S_2) for agent b. Note that generation and certificate issue are part of a single rule. This allows us to eliminate the need for a certification command, for which deciding the key authenticity could raise a problem.

$$\begin{aligned}
&P_e(h_e^{\alpha}(N_1, N_2, inv(Y_k), \text{privCertSign}, i_1, S_{e,1}, S_{e,2})),\\
&\qquad P_e(h_e(N_2, \mathcal{C}(N_2, N_1, N_{cert}, Y_k, \text{pubCertVerif}, i_1, S_{e,1}, S_{e,2}))), i_2, S_1, S_2, b \overset{N_3, N_4, X_k}{\Longrightarrow}\\
&\qquad P_e(h_e^{g}(N_3, N_4, inv(X_k), \text{privDecSign}, i_2, S_{e,1} \cup S_{e,2} \cup S_1 \cup \{e\}, \{b, e\} \cup S_2)),\\
&P_e(\Sigma(\mathcal{C}(N_4, N_3, N_2, X_k, \text{pubEncVerif}, i_2, S_{e,1} \cup S_{e,2} \cup S_1 \cup \{e\}, \{b, e\} \cup S_2), inv(Y_k))), \qquad \textbf{(Asym Gen)}
\end{aligned}$$

with $e \in S_{e,2}$, $i_1, i_2 \in I_{>0}$, $\alpha \in \{r, g\}$ on condition that $i_2 < i_1$.

Asymmetric encryption with signature This API command enables an agent b, owner of a private handle pointing to an asymmetric key Y_k, to encrypt and sign a payload for agent c, provided b has an integrity handle for a public key X_k of c. As in the symmetric case, handles in payload $m_1, \ldots, m_n$ are translated into real values and attributes. Encryption and signature needs to be an atomic command to enable the device to control what can be signed.

$$\begin{aligned}
&P_b(h_b^{\alpha}(N_1, N_2, inv(Y_k), \text{privDecSign}, i_b, S_{b,1}, S_{b,2}),\\
&\qquad P_b(h_b(N_3, \mathcal{C}(N_3, N_4, N_5, X_k, \text{pubEncVerif}, i_c, S_{c,1}, S_{c,2}))),\\
&P_b(m_1), \ldots, P_b(m_n) \Longrightarrow P_b(\{m'_1, \ldots, m'_n\}_{X_k}), P_b(\Sigma(\{m'_1, \ldots, m'_n\}_{X_k}, inv(Y_k))),\\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \textbf{(Asym SignEncrypt)}
\end{aligned}$$

with $i_b, i_c \in I_{>0}$, $c \in S_{c,2}$, $m_j, m'_j \in \mathsf{Msg}$ and for $j = 1, \ldots, n$:

- if $m_j = h_b^{\alpha}(N_j, N'_j, X_{k,j}, T_j, i_j, S_{j,1}, S_{j,2})$ with $X_{k,j} \in \mathsf{Keyv} \cup \mathsf{Noncev}$ then :
  - if $i_b, i_c \in I_2$, $(i_j, S_{j,1}, S_{j,2}) \prec (i_b, S_{b,1}, S_{b,2})$ and $(i_j, S_{j,1}, S_{j,2}) \prec (i_c, S_{c,1}, S_{c,2})$ then $m'_j = N_j, N'_j, X_{k,j}, T_j, i_j, S_{j,1}, S_{j,2}$;
  - else $m'_j = \varnothing$.
- else $m'_j = \mathsf{nhdl}(m_j)$.

Asymmetric decryption with signature verification The following rule allows for decryption by the agent b of an authenticated ciphertext, using an integrity handle pointing to a public key Y_k to verify the signature and a handle pointing to a key $inv(X_k)$ to decrypt the ciphertext.

$$\begin{aligned}
&P_b(h_b(N_1, \mathcal{C}(N_1, N_2, N_3, Y_k, \text{pubEncVerif}, i_c, S_{c,1}, S_{c,2}))),\\
&\qquad P_b(h_b^{\alpha}(N_4, N_5, inv(X_k), \text{privDecSign}, i_b, S_{b,1}, S_{b,2})),\\
&\qquad P_b(\{m_1, \ldots, m_n\}_{X_k}), P_b(\Sigma(\{m_1, \ldots, m_n\}_{X_k}, inv(Y_k)))\\
&\Longrightarrow P_b(m'_1), \ldots, P_b(m'_n), \qquad \textbf{(Asym VerifDecrypt)}
\end{aligned}$$

with $i_b, i_c \in I_{>0}$, $m_j, m'_j \in \mathsf{Msg}$ and for $j = 1, \ldots, n$:

- if $m_j = N_j, N'_j, X_{k,j}, T_j, i_j, S_{j,1}, S_{j,2}$ then

[1]We require that both parts of the certification key exist in the creating agent's secure hardware. This is not a compulsory security constraint, in the sense that a few modifications can be performed in the rules and proof to get rid of it. However, it seems reasonable in practice to perform such a verification.

  - if $i_b, i_c \in I_2$, $(i_j, S_{j,2}, S_{j,2}) \prec (i_b, S_{b,1}, S_{b,2})$ and $(i_j, S_{j,2}, S_{j,2}) \prec (i_c, S_{c,1}, S_{c,2})$ then $m'_j = h^r_b(N_j, N'_j, X_{k,j}, T_j, i_j, S_{j,1}, S_{j,2})$;
  - else $m'_j = \varnothing$.

- if $m_j = \mathsf{nhdl}(t_j)$ for $t_j \in \mathsf{Msg}$ then $m'_j = t_j$.

Certification key generation Given a certification key pair under handles, this rule allows agent e to generate a certification key pair $(X_k, inv(X_k))$ for agent b. As for asymmetric generation, generation and certificate issue are part of an atomic call. It eliminates the need for a certification command, for which deciding the key authenticity could raise a problem.

$$\begin{aligned}
&P_e(h^\alpha_e(N_1, N_2, inv(Y_k), \text{privCertSign}, i_e, S_{e,1}, S_{e,2})),\\
&\qquad P_e(h_e(N_2, \mathcal{C}(N_2, N_1, N_{cert}, Y_k, \text{pubCertVerif}, i_e, S_{e,1}, S_{e,2}))), i_b, S_1, S_2 \overset{N_3, N_4, X_k}{\Longrightarrow}\\
&\qquad P_e(h^g_e(N_3, N_4, inv(X_k), \text{privCertSign}, i_b, S_{e,1} \cup S_{e,2} \cup S_1 \cup \{e\}, \{e, b\} \cup S_2)),\\
&P_e(\Sigma(\mathcal{C}(N_4, N_3, N_2, X_k, \text{pubCertVerif}, i_b, S_{e,1} \cup S_{e,2} \cup S_1 \cup \{e\}, \{e, b\} \cup S_2), inv(Y_k))),
\end{aligned}$$

(**Cert Gen**)

with $e \in S_{e,2}$, $i_e \in I_2$ and $i_b \leq i_e$.

Verification of a certificate This rule allows an agent b, given an integrity handle pointing to a verification key and a pre-certificate signed by the matching certification key, to create the suitable integrity handle. For $\Theta \in \{\text{EncVerif}, \text{CertVerif}\}$,

$$\begin{aligned}
&P_b(\Sigma(\mathcal{C}(N_1, N_2, N_3, X_k, pub\Theta, i_c, S_{c,1}, S_{c,2}), inv(Y_k))),\\
&\qquad P_b(h_b(N_3, \mathcal{C}(N_3, N_4, N_5, Y_k, \text{pubCertVerif}, i_e, S_{e,1}, S_{e,2}))) \Longrightarrow\\
&P_b(h_b(N_1, \mathcal{C}(N_1, N_2, N_3, X_k, pub\Theta, i_c, S_{c,1}, S_{c,2}))),
\end{aligned}$$

(**Cert Verif**)

with $i_c, i_e \in I_{>0}$ and $(i_c, S_{c,1}, \varnothing) \prec (i_e, S_{e,1} \cup S_{e,2}, \varnothing)$.

3.3 Example

In Figure 3 we show the 'before' and 'after' states for three agents using the API in a typical configuration. In the 'before' state, there are no shared secrets. Alice and Bob both have accepted a copy of the CA's public key certificate and placed it under an integrity handle (identifiers 5 and 8 respectively) and they have generated their own public-private keypairs. The CA has accepted public key certificates for each of these pairs (identifiers 3 and 4). Here we are using integers to label key levels, arbitrarily assigning the long term keys the level 3. Public keys are level 0.

To establish a shared secret, Alice and Bob first need to accept each others public key certificates. This can be done by requesting them from the CA. The CA uses the AsymEncryptSign command to sign the (public) message containging the certificate. Now Alice and Bob can use the certificate verification command to accept the certificates, generating handles 11 and 12.

Now either Alice can generate a symmetric key (handle 13) and send it to Bob using AsymEncryptSign. Bob will use AsymDecryptVerify and accept the key (handle 14). Now Alice and Bob can exchange messages using the new symmetric key. Note that the new symmetric key is confidential between Alice and Bob, hence has a confidentiality control set S_2 containing only these identifiers, but for integrity it has inherited the dependence on the CA, hence S_1 contains the set of agents CA, Alice and Bob.

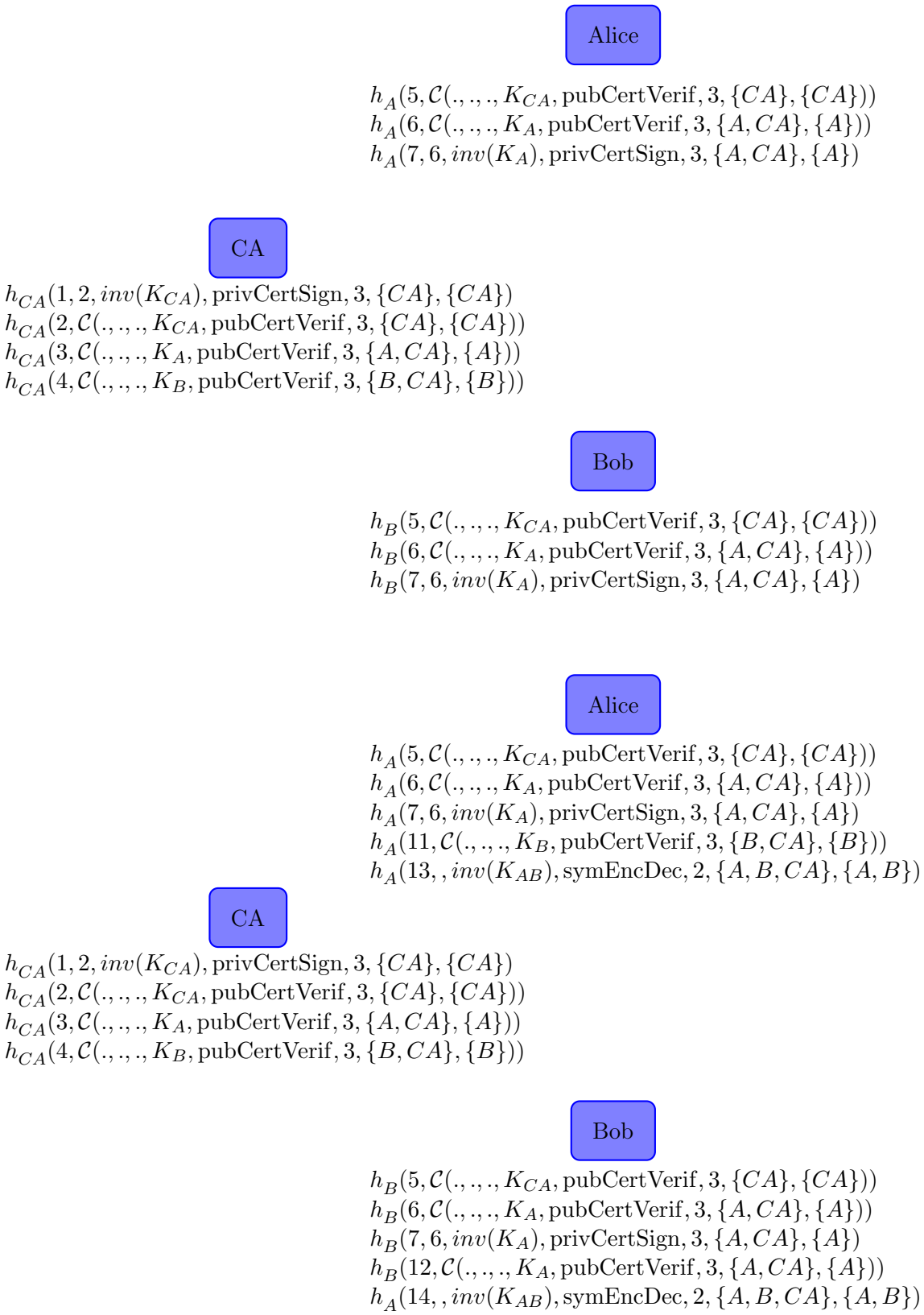


Figure 3: Operation of the API. See 3.3 for narration.

3.4 Security rationale

Below we will formally prove security properties for our design, but first we discuss the design features that prevent it from suffering from the kinds of attacks seen in the literature [2, 3, 5]. First, a vital step in several attacks against security APIs is the ability to change the value of the attributes of a key over time or from one component to another. Our design, much as the Cortier-Steel design, prevents such attempts in two ways. Firstly, the attributes of a key are set once and for all when it is generated or imported onto a device. Secondly, when transporting keys, we export all attributes along with the value of the key and protect their integrity.

In the specification of the widely used API PKCS#11, a key can posess conflicting attributes, such as 'Wrap' and 'Decrypt' leading to attacks [6]. These attacks are not possible in our design because of the distinction between the way keys and data are tagged for encryption: either as a concatenation of key and attributes or encapsulated in a container nhdl. In an implementation of our design, a suitable tagging scheme should be used to ensure this distinction.

Key conjuring, i.e. the ability of the adversary to generate any number of (possibly related) keys on the device, is critical to a number of attacks [2]. Careful design of the decrypt command prevents this. The security proof incudes an enumeration of the terms which the adversary can successfully submit to a decryption request (see (**Sign**) and (**SymEnc**)). Roughly, suitable terms are either wrapped under compromised keys or result from an honest use of the encrypt command.

4 Security of the API in the symbolic model

4.1 Model of security

In this section, we describe the capacity of the attacker in the spirit of Dolev and Yao [9], as formalized in [1].

4.1.1 Computation of new terms

We denote by $\mathsf{INTRUDER}$ the set of rules which allow the attacker to build new terms from the ones that it has already.

Creation/destruction of pairs Let $m_1, m_2 \in \mathsf{Msg}$, we have:

- $P_{\mathsf{int}}(m_1), P_{\mathsf{int}}(m_2) \Rightarrow P_{\mathsf{int}}(< m_1, m_2 >)$
- $P_{\mathsf{int}}(< m_1, m_2 >) \Rightarrow P_{\mathsf{int}}(m_1), P_{\mathsf{int}}(m_2)$

Symmetric encryption/decryption let $X_k \in \mathsf{Keyv}$, $m_1, \ldots, m_n \in \mathsf{Msg}$, we have :

- $P_{\mathsf{int}}(X_k), P_{\mathsf{int}}(m_1), \ldots, P_{\mathsf{int}}(m_n) \Rightarrow P_{\mathsf{int}}(\{\!|m_1, \ldots, m_n|\!\}_{X_k})$
- $P_{\mathsf{int}}(X_k), P_{\mathsf{int}}(\{\!|m_1, \ldots, m_n|\!\}_{X_k}) \Rightarrow P_{\mathsf{int}}(m_1), \ldots, P_{\mathsf{int}}(m_n)$

Asymmetric encryption/decryption Let $X_k \in \mathsf{Keyv}$, $m_1, \ldots, m_n \in \mathsf{Msg}$, we have :

- $P_{\mathsf{int}}(X_k), P_{\mathsf{int}}(m_1), \ldots, P_{\mathsf{int}}(m_n) \Rightarrow P_{\mathsf{int}}(\{m_1, \ldots, m_n\}_{X_k})$
- $P_{\mathsf{int}}(inv(X_k)), P_{\mathsf{int}}(\{m_1, \ldots, m_n\}_{X_k}) \Rightarrow P_{\mathsf{int}}(m_1), \ldots, P_{\mathsf{int}}(m_n)$

Signature Let $X_k \in \mathsf{Keyv}$, $m \in \mathsf{Msg}$, we have :

- $P_{\mathsf{int}}(X_k), P_{\mathsf{int}}(m) \Rightarrow P_{\mathsf{int}}(\Sigma(m, X_k))$
- $P_{\mathsf{int}}(\Sigma(m, X_k)) \Rightarrow P_{\mathsf{int}}(m)$

Creation/destruction of container of regular messages Let $m \in \mathsf{Msg}$, we have :

- $P_{\mathsf{int}}(m) \Rightarrow P_{\mathsf{int}}(\mathsf{nhdl}(m))$
- $P_{\mathsf{int}}(\mathsf{nhdl}(m)) \Rightarrow P_{\mathsf{int}}(m)$

The transitive reflexive closure of the preceding rules can be interpreted as the set of terms that an attacker can deduce from its knowledge at a certain state. In the following, we say that m is deducible from a set of terms T, that we denote by $T \vdash m$, if starting from the state $\mathcal{S}$ such that $\mathcal{S}_{\mathsf{int}} = T$ and for all $a \in \mathsf{Agent}$, $\mathcal{S}_a = \emptyset$, there exists a state $\mathcal{S}'$ such that $\mathcal{S} \Longrightarrow^*_{\mathsf{INTRUDER}} \mathcal{S}'$ and $m \in \mathcal{S}'_{\mathsf{int}}$. In the following, we will do the following abuse of notation : if t is a term and $\mathcal{S}$ is a state, we write $t \in \mathcal{S}$ (resp. $\mathcal{S} \vdash t$) if $t \in \cup_{b \in \mathsf{Agent} \cup \{\mathsf{int}\}} \mathcal{S}_b$ (resp. if $\cup_{b \in \mathsf{Agent} \cup \{\mathsf{int}\}} \mathcal{S}_b \vdash t$).

4.1.2 Control of the network and corruption

Control of the network Let a be an agent m a message.

- $P_a(m) \Rightarrow P_{\mathsf{int}}(m)$
- $P_{\mathsf{int}}(m) \Rightarrow P_a(m)$

In the following, we suppose given a set H a honest agents.

Corruption of a device Let a be a dishonest agent (i.e. $a \notin H$) and

$$h^\alpha_a(N_1, N_2, m, T, i, S_1, S_2),$$

a handle then

$$P_a(h^\alpha_a(N_1, N_2, m, T, i, S_1, S_2)) \Rightarrow P_{\mathsf{int}}(m).$$

4.2 Initial states

Secure hardware devices are set up in a controlled environment. We impose a few requirements on the state of a device at the end of the initialisation process. We consider states satisfying these conditions as initial states for the transition system defined by the set of API and adversary rules. These requirements seem realistic in pratice and allow us to start from states compatible with the security policy. In the initial states, we assume that the attacker knows some public information like the set of key levels and the set of agents.

Definition 1. A state $\mathcal{S}_0$ is said to be *initial* if it satisfies the following hypotheses :

1. the set of terms known by the agents and the intruder are atomic : for all $a \in \mathsf{Agent} \cup \{\mathsf{int}\}$, $\mathcal{S}_a \subseteq \mathsf{Handle} \cup \mathsf{Key} \cup \mathsf{Nonce} \cup \mathsf{Agent} \cup \mathcal{T} \cup I \cup \mathscr{S}$ and moreover $\mathcal{T} \cup I \cup \mathscr{S} \subseteq \mathcal{S}_{\mathsf{int}}$.
2. all terms stored under handles are secret : for $a \in \mathsf{Agent}$, if $h^\alpha_a(N_1, N_2, m, T, i, S_1, S_2) \in \mathcal{S}_a$ then for $b \in \mathsf{Agent} \cup \{\mathsf{int}\}$, $m \notin \mathcal{S}_b$.
3. all key handles known by an agent point to an atomic element : for $a \in \mathsf{Agent}$, if $h^\alpha_a(N_1, N_2, m, T, i, S_1, S_2) \in \mathcal{S}_a$ then $m \in \mathsf{Key} \cup \mathsf{Nonce}$.
4. the owner of a key handle is in the set of legitimate users. More precisely, we impose that for all $a \in \mathsf{Agent}$, if $h^\alpha_a(N_1, N_2, m, T, i, S_1, S_2) \in \mathcal{S}_a$ then $a \in S_2$.

5. any public key certificate under handle corresponds to a private key stored by a rightful agent: $\forall b \in \mathsf{Agent}$, if $h_b(N_1, \mathcal{C}(N_1, N_2, N_3, X_k, \mathsf{pub}\Theta, i, S_1, S_2)) \in \mathcal{S}_b$, then there exists $a \in S_2$ so that

$$h_a^\alpha(N_2, N_1, inv(X_k), \mathsf{priv}\Theta', i, S_1, S_2) \in \mathcal{S}_a,$$

with $(\Theta, \Theta') \in \{(\text{EncVerif}, \text{DecSign}), (\text{CertVerif}, \text{CertSign})\}$.

6. the key handles form a *coherent* set: for all $a, a' \in \mathsf{Agent}$, $h_a^\alpha(N_1, N_2, m, T, i, S_1, S_2) \in \mathcal{S}_a$ and $h_{a'}^{\alpha'}(N_1', N_2', m, T', i', S_1', S_2') \in \mathcal{S}_{a'}$ then $N_1 = N_1', T = T', i = i', N_2 = N_2', S_1 = S_1'$ and $S_2 = S_2'$.

We can now define the set of states for which we can prove a security property.

Definition 2. We say that a state $\mathcal{S}$ is *accessible* from an initial state $\mathcal{S}_0$ if it is reachable by applying a finite number of times the rules of the set API, $\mathsf{INTRUDER}$ and $\mathsf{CONTROL}$ to $\mathcal{S}_0$, i.e. if $\mathcal{S}_0 \Rightarrow^*_{\mathsf{API} \cup \mathsf{CONTROL} \cup \mathsf{INTRUDER}} \mathcal{S}$.

4.3 Security properties and sketch of proof

The security of the API can be expressed in the following way: given a state $\mathcal{S}$, secret data of honest agents should not be known to the intruder. But we would also like to ensure that this data is only used by rightful agents. Secret data of honest users are messages $m \in \mathsf{Msg}$ for which there exists a handle of the form $h_a^\alpha(., ., m, ., ., S_1, S_2)$ with $a \in H$ and $S_1, S_2 \subseteq H$. As the set of legitimate users of m is S_2, the property that we want to prove is formalized as:

$$\forall a \in H, \forall m \in \mathsf{Msg}, \forall i \in I_{>0}, \forall \alpha \in \{r, g\}, \forall S_1, S_2 \subseteq H,\ \mathcal{S} \vdash h_a^\alpha(., ., m, ., i, S_1, S_2) \Rightarrow \mathcal{S} \nvdash m \text{ and } a \in S_2 \quad \textbf{(Sec)}$$

We can now give the principal result of this paper, stating the security of our API (**Sec**) if it is correctly initialised.

Theorem 1 (Confidentiality of data under handles)**.** Let $\mathcal{S}_0$ be an initial state and $\mathcal{S}$ be an accessible state from $\mathcal{S}_0$. Then $\mathcal{S}$ satisfies the property (**Sec**).

Proof. We present in this section a sketch of proof, to provide some intuitions about the way it works. Details can be found in section 5. We first begin to consider a more powerful attacker which has access to all the values stored in compromised hardware as well as to all the messages m associated to handles of the form $h_a^\alpha(., ., m, ., ., S_1, S_2)$ where $S_1, S_2 \subsetneq H$ even if a is honest. The underlying idea is that the classic adversary can easily learn these terms anyway. Moreover, this extension allows to ensure stability of the intruder's knowledge when applying rules from set $\mathsf{INTRUDER} \cup \mathsf{CONTROL}$.

It yields a generalized deduction definition: we write that $\mathcal{S} \vdash^* t$ when $\cup_{b \in \mathsf{Agent} \cup \{\mathsf{int}\}} \mathcal{S}_b \cup \{m, N_1, N_2 | h_a^\alpha(N_1, N_2, m, ., ., S_1, S_2) \in \mathcal{S}, S_1 \subsetneq H \text{ or } S_2 \subsetneq H, a \in \mathsf{Agent}\} \cup \{m, N_1, N_2 | h_a^\alpha(N_1, N_2, m, ., ., ., .) \in \mathcal{S}, a \notin H\} \cup \{m | h_a(., m) \in \mathcal{S}\} \vdash t$.

We then consider a stronger version of the property (**Sec**):

$$\forall a \in H, \forall m \in \mathsf{Msg}, \forall i \in I_{>0}, \forall \alpha \in \{r, g\}, \forall S_1, S_2 \subseteq H,\ \mathcal{S} \vdash^* h_a^\alpha(., ., m, ., i, S_1, S_2) \Rightarrow \mathcal{S} \nvdash^* m, a \in S_2 \text{ and } m \in \mathsf{Key} \cup \mathsf{Nonce}. \quad \textbf{(Sec}^*\textbf{)}$$

Intuitively, the property (**Sec***) means that the values stored in the handles of honest agents are always of type Key or Nonce and are not deducible even with the extended deduction rule $\vdash^*$. It is clear that in order to prove the theorem, it is enough to prove the same statement with the stronger version of the property (**Sec**).

In the next section, we prove by induction that the property (**Sec***) is invariant by the API rules. Intuitively, all keys deducible from an initial state in the extended sense are compromised keys (in the

sense that there is an agent owning the handle or appearing in the control sets that is dishonest). The idea behind the induction is to show that this remains true after each API function call. (We recall that stability by $\mathsf{INTRUDER} \cup \mathsf{CONTROL}$ follows from the use of the extended deducibility notion.)

To prove this, we introduce four invariants[2] to show the inability of an adversary to create interesting wrapped secret keys or signed certificates allowing him to extract secret data from a security device. The first invariant, denoted (**SymEnc**), states that the only well-formed symmetric encryption terms that an adversary can build are either encrypted under a compromised key, or resulting from an honest and well-formed request to the symmetric encryption command. If the latter is true, we know that the key hierarchy is preserved because of the design of the encryption command. Otherwise, potential keys appearing under encryption were already deducible (in the extended sense) by the adversary. More formally, the invariant can be written as follows:

$$\begin{array}{l}
\forall u, k \in \mathsf{Msg}, \mathcal{S} \vdash^* \{\!|u|\!\}_k \Rightarrow \mathcal{S} \vdash^* k \textbf{ or} \\
\qquad \exists i \in I_{>0}, \exists S_1, S_2 \subset H \ \exists a \in S_2 \text{ such that } \mathcal{S} \vdash^* h_a^i(.,.,k,.,i,S_1,S_2) \\
\qquad\quad \exists u'_1, \ldots, u'_p \in \mathsf{Msg} \text{ such that } u = u'_1, \ldots, u'_p \text{ with } \forall j = 1 \ldots p, \\
\textbf{either } \exists m_j \in \mathsf{Key} \cup \mathsf{Nonce}, \exists T_j \in T, \exists (i_j, S_{j,1}, S_{j,2}) \in (I \times \mathscr{S} \times \mathscr{S}), \exists (N_{j,1}, N_{j,2}) \in \mathsf{Nonce}^2 \text{ such that} \\
\qquad u'_j = N_{j,1}, N_{j,2}, m_j, T_j, i_j, S_{j,1}, S_{j,2},\ (i_j, S_{j,1}, S_{j,2}) \prec (i, S_1, S_2) \text{ and} \\
\qquad\quad \mathcal{S} \vdash^* h_a(N_{j,1}, N_{j,2}, m_j, T_j, i_j, S_{j,1}, S_{j,2}) \\
\qquad\quad\quad \textbf{or } \exists m_j \in \mathsf{Msg} \text{ such that } m'_j = \mathsf{nhdl}(m_j) \text{ and } \mathcal{S} \vdash^* m_j. \quad \textbf{(SymEnc)}
\end{array}$$

The next invariant states that the only asymmetric encryption terms deducible from a state accessible from an initial state either have a payload deducible by the attacker, or result from an honest and well-formed request to the asymmetric encryption command.

$$\begin{array}{l}
\forall u, K \in \mathsf{Msg}, \mathcal{S} \vdash^* \{u\}_K \Rightarrow \mathcal{S} \vdash^* u \textbf{ or} \\
\quad \exists S_{b,1}, S_{b,2} \subset \mathsf{Agent}, \exists b \in S_{b,2}, \exists i_b \in I \text{ such that } \mathcal{S} \vdash^* h_b(.,.,k, \text{privDecSign}, i_b, S_{b,1}, S_{b,2}) \\
\qquad \exists S_{c,1}, S_{c,2} \subset \mathsf{Agent}, \exists i_c \in I,\ \exists K \in \mathsf{Key} \text{ such that} \\
\qquad\quad \mathcal{S} \vdash^* h_b(., \mathcal{C}(.,.,.,K, \text{pubEncVerif}, i_c, S_{c,1}, S_{c,2})) \\
\qquad \exists u'_1, \ldots, u'_p \in \mathsf{Msg} \text{ such that } u = u'_1, \ldots, u'_p \text{ with } \forall j = 1 \ldots p, \\
\textbf{either } \exists m_j \in \mathsf{Key} \cup \mathsf{Nonce}, \exists T_j \in T, \exists (i_j, S_{j,1}, S_{j,2}) \in (I \times \mathscr{S} \times \mathscr{S}), \exists (N_{j,1}, N_{j,2}) \in \mathsf{Nonce}^2 \text{ such that} \\
u'_j = N_{j,1}, N_{j,2}, m_j, T_j, i_j, S_{j,1}, S_{j,2},\ (i_j, S_{j,1}, S_{j,2}) \prec (i_b, S_{b,1}, S_{b,2}), (i_j, S_{j,1}, S_{j,2}) \prec (i_c, S_{c,1}, S_{c,2}) \text{ and} \\
\mathcal{S} \vdash^* h_b(N_{j,1}, N_{j,2}, m_j, T_j, i_j, S_{j,1}, S_{j,2}) \textbf{ or } \exists m_j \in \mathsf{Msg} \text{ such that } u'_j = \mathsf{nhdl}(m_j) \text{ and } \mathcal{S} \vdash^* m_j \\
\textbf{(AsymEnc)}
\end{array}$$

We need a similar invariant for signed terms the adversary is able to obtain. Once again, the idea is to show that if the adversary has not compromised the signing or certifying key, the signed terms he can build follow directly from an honest API call. The invariant here is slightly more involved since we have to deal with both the issue of certificates when generating asymmetric keys and asymmetric wrapping commands. This is formalized in invariant (**Sign**) as follows:

[2]Our submitted paper mentioned only three invariants. The fourth was added to enable us to weaken the restrictions on the asymmetric encryption and decryption commands to make them more widely applicable.

$$\forall u, k \in \mathsf{Msg}, \mathcal{S} \vdash^* \Sigma(u,k) \Rightarrow \mathcal{S} \vdash^* k \textbf{ or}$$
$$\exists b \in \mathsf{Agent}, \exists e \in \mathsf{Agent}, \exists i_1, i_2 \in I, \exists N_2, N_3, N_4 \in \mathsf{Nonce}, \exists S_1', S_2', S_1, S_2 \subset \mathsf{Agent}, \exists X_k \in \mathsf{Keyv} \text{ such that}$$
$$u = \mathcal{C}(N_4, N_3, N_2, X_k, \mathsf{pub}\Theta, i_2, S_1 \cup \{e\}, S_2 \cup \{b, e\}) \text{ and}$$
$$\mathcal{S} \vdash^* h_{\dot{e}}(.,.,k, \text{privCertSign}, i_1, S_1', S_2') \text{ with } S_1' \cup S_2' \subset S_1,\ e \in S_2' \subset \mathsf{Agent}, i_2 < i_1$$
$$\textbf{or}$$
$$\exists S_{b,1}, S_{b,2} \subset \mathsf{Agent}, \exists b \in S_{b,2}, \exists i_b \in I \text{ such that } \mathcal{S} \vdash^* h_{\dot{b}}(.,.,k, \text{privDecSign}, i_b, S_{b,1}, S_{b,2})$$
$$\exists S_{c,1}, S_{c,2} \subset \mathsf{Agent}, \exists i_c \in I,\ \exists K \in \mathsf{Key} \text{ such that}$$
$$\mathcal{S} \vdash^* h_b(., \mathcal{C}(.,.,.,K, \text{pubEncVerif}, i_c, S_{c,1}, S_{c,2}))$$
$$\exists u_1', \ldots, u_p' \in \mathsf{Msg} \text{ such that } u = \{u_1', \ldots, u_p'\}_K \text{ with } \forall j = 1 \ldots p,$$
$$\textbf{either } \exists m_j \in \mathsf{Key} \cup \mathsf{Nonce}, \exists T_j \in T, \exists (i_j, S_{j,1}, S_{j,2}) \in (I \times \mathscr{S} \times \mathscr{S}), \exists (N_{j,1}, N_{j,2}) \in \mathsf{Nonce}^2 \text{ such that}$$
$$u_j' = N_{j,1}, N_{j,2}, m_j, T_j, i_j, S_{j,1}, S_{j,2},\ (i_j, S_{j,1}, S_{j,2}) \prec (i_b, S_{b,1}, S_{b,2}), (i_j, S_{j,1}, S_{j,2}) \prec (i_c, S_{c,1}, S_{c,2})) \text{ and}$$
$$\mathcal{S} \vdash^* h_{\dot{b}}(N_{j,1}, N_{j,2}, m_j, T_j, i_j, S_{j,1}, S_{j,2})$$
$$\textbf{or } \exists m_j \in \mathsf{Msg} \text{ such that } u_j' = \mathsf{nhdl}(m_j) \text{ and } \mathcal{S} \vdash^* m_j \quad \textbf{(Sign)}$$

for $\Theta \in \{\text{EncVerif}, \text{CertSign}\}$.

Intuitively, the property (**Cert**) means that if a certificate is under an integrity handle with control sets S_1, S_2 containing only honest users, then there exists a private key handle associated to this certificate the attributes of which are coherent with that of the certificate.

$$\forall a \in H, \forall N_1, N_2, N_3 \in \mathsf{Nonce}, \forall i \in I_{>0}, \forall S_1, S_2 \subseteq H \text{ with}$$
$$\mathcal{S} \ \vdash^* \ h_a(N_1, \mathcal{C}(N_1, N_2, N_3, X_k, pub\Theta, i, S_1, S_2)) \ \Rightarrow \ \exists b \in S_2 \text{ such that}$$
$$\mathcal{S} \ \vdash^* \ h_b^{\alpha}(N_2, N_1, inv(X_k), priv\Theta', i, S_1, S_2). \quad \textbf{(Cert)}$$

where $(\Theta, \Theta') \in \{(\text{EncVerif}, \text{DecSign}), (\text{CertVerif}, \text{CertSign})\}$.

To finish the sketch of proof, we remark moreover that from its definition, an initial state satisfies the properties (**Sec***), (**SymEnc**), (**AsymEnc**) (**Cert**), (**Sign**).

□

5 Detailed proof

In this section, we provide a detailed proof of Theorem 1. We want to show that the properties (**Sec***), (**SymEnc**), (**AsymEnc**), (**Cert**), (**Sign**) are invariant by application of the rules $\mathsf{API} \cup \mathsf{INTRUDER} \cup \mathsf{CONTROL}$. We remark that transitions of set API can create fresh handles, symmetric and asymmetric encrypted messages, or signatures. The following technical lemma proves useful repeatedly during the proof. It states that from handles, encrypted messages the decryption key of which is unknown to him and from signatures, the attacker can not create new terms which are useful for its purpose.

Lemma 1. Let $k_1, k_2 \in \mathsf{Key}$ and let $t_1, t_2 \in \mathsf{Msg}$. We denote by f the symmetric or asymmetric encryption function. Let S and S' be sets of terms such that

$$S' \subseteq S \cup \{f(t_1, k_1)\} \cup \ \{\Sigma(t_2, k_2)\} \cup \mathsf{Hdls} \cup \mathsf{NK}$$

with NK a set of elements of $\mathsf{Key} \cup \mathsf{Nonce} \cup \mathsf{Agent} \cup \mathcal{T} \cup I \cup \mathscr{S}$ which do not appear in $S \cup \{f(t_1, k_1)\} \cup \{\Sigma(t_2, k_2)\}$ and $\mathsf{Hdls} \subseteq \mathsf{Handle}$ a set of handles which are either

- of the form $h_a^{\alpha}(.,.,k,.,.,S_1,S_2)$ with $S_1 \cup S_2 \cup \{a\} \subseteq H$, or $S_1 \cup S_2 \cup \{a\} \nsubseteq H$, and then we suppose that $S \vdash^* k$ or that $k \in \mathsf{NK}$;

- or of the form $h_a(., N)$ with $a \in \mathsf{Agent}$ and $S \vdash^* N$ or $\mathsf{NK} \vdash N$.

We suppose moreover that $S \cup \{f(t_1, k_1)\} \cup \mathsf{NK} \vdash^* t_2$ and that k_1' - with $k_1' = k_1$ if k_1 is a symmetric key (resp. $k_1' = inv(k_1)$ if k_1 is an asymmetric key) - and k_2 are not deducible from $\mathcal{S}$, that is to say that $S \not\vdash^* k_1'$, $S \not\vdash^* k_2$, $k_1', k_2 \notin \mathsf{NK}$. Let $u \in \mathsf{Agent} \cup \mathsf{Nonce} \cup \mathsf{Key} \cup \mathsf{Handle}$ be an atomic message. We have :

$$S' \vdash^* u \text{ if and only if } S \vdash^* u \text{ or } u \in \mathsf{Hdls} \text{ or } u \in \mathsf{NK}. \tag{1}$$

Moreover, let $v \in \mathsf{Msg}$ and $w \in \mathsf{Key}$, then we have

$$S' \vdash^* \{\!|v|\!\}_w \text{ and } S' \not\vdash^* w \text{ only if } S \vdash^* \{\!|v|\!\}_w \text{ or } \{\!|v|\!\}_w = f(t_1, k_1), \tag{2}$$

$$S' \vdash^* \{v\}_w \text{ and } S' \not\vdash^* v \text{ only if } S \vdash^* \{v\}_w \text{ or } \{v\}_w = f(t_1, k_1), \tag{3}$$

$$S' \vdash^* \Sigma(v, w) \text{ and } S' \not\vdash^* w \text{ only if } S \vdash^* \Sigma(v, w) \text{ or } \Sigma(v, w) = \Sigma(t_2, k_2). \tag{4}$$

Remark 1. In the statement, when we say that an element of NK does not appear in $S \cup \{f(t_1, k_1)\} \cup \{\Sigma(t_2, k_2)\}$ this means in particular that if $inv(k) \in \mathsf{NK}$ then k does not appear in $S \cup \{f(t_1, k_1)\} \cup \{\Sigma(t_2, k_2)\}$.

Proof. We let $S_\infty = \{u | S \vdash^* u\}$, $S_0' = S' \cup S_\infty \subseteq S_\infty \cup \{f(t_1, k_1)\} \cup \{\Sigma(t_2, k_2)\} \cup \mathsf{Hdls} \cup \mathsf{NK}$ and we consider the sets $(S_i')_{i \in \mathbb{N}}$ defined inductively by:

$$S_{i+1}' = S_i' \cup \{u | S_i' \cup \Delta_i \Rightarrow_R u, R \in \mathsf{INTRUDER}\},$$

where $\Delta_i = \{m, N_1, N_2 | h_a^\alpha(N_1, N_2, m, ., ., S_1, S_2) \in S_i', S_1 \subsetneq H \text{ or } S_2 \subsetneq H, a \in \mathsf{Agent}\} \cup \{m, N_1, N_2 | h_a^\alpha(N_1, N_2, m, ., ., ., .) \in S_i', a \notin H\} \cup \{m | h_a(., m) \in S_i'\}$. On the other side, we let $F_i' = S_i' - S_\infty$. We show by induction that

$$u \in (\mathsf{Agent} \cup \mathsf{Nonce} \cup \mathsf{Key} \cup \mathsf{Handle}) \cap F_i' \Rightarrow u \in \mathsf{Hdls} \cup \mathsf{NK}, \tag{5}$$

$$\forall v \in \mathsf{Msg}, \forall w \in \mathsf{Key}, \Sigma(v, w) \in S_i' \text{ and } w \notin S_i' \Rightarrow \Sigma(v, w) \in S_\infty \text{ or } \Sigma(v, w) = \Sigma(t_2, k_2), \tag{6}$$

$$\forall v \in \mathsf{Msg}, \forall w \in \mathsf{Key}, \{\!|v|\!\}_w \in S_i' \text{ and } w \notin S_i' \Rightarrow \{\!|v|\!\}_w \in S_\infty \text{ or } \{\!|v|\!\}_w = f(t_1, k_1), \tag{7}$$

$$\forall v \in \mathsf{Msg}, \forall w \in \mathsf{Key}, \{v\}_w \in S_i' \text{ and } v \notin S_i' \Rightarrow \{v\}_w \in S_\infty \text{ or } \{v\}_w = f(t_1, k_1). \tag{8}$$

At the same time, we prove that S_i' is included in the union of

$$A = S_\infty \cup f(t_1, k_1) \cup \Sigma(t_2, k_2) \cup \mathsf{Hdls} \cup \mathsf{NK}, \tag{9}$$

$$B_i = < m_1, m_2 > \text{ with } m_1, m_2 \in S_{i-1}', \tag{10}$$

$$C_i = \{\!|m_1, \ldots, m_n|\!\}_K \text{ with } m_1, \ldots, m_n, K \in S_{i-1}', \tag{11}$$

$$D_i = \{m_1, \ldots, m_n\}_K \text{ with } m_1, \ldots, m_n, K \in S_{i-1}', \tag{12}$$

$$E_i = \Sigma(m_1, K) \text{ with } m_1, K \in S_{i-1}', \tag{13}$$

$$F_i = \mathsf{nhdl}(m) \text{ with } m \in S_{i-1}'. \tag{14}$$

It is clear that the induction hypothesis is verified for S_0' (by setting $S_{-1}' = \emptyset$). Suppose that the hypothesis are true for S_i', we prove the hypothesis for S_{i+1}'.

We begin by remarking that because of the hypothesis made on the handles of Hdls, we have $\Delta_i \subset S_\infty \cup \mathsf{NK}$. Thus, we have $S_{i+1}' = S_i' \cup \{u | S_i' \Rightarrow_R u, R \in \mathsf{INTRUDER}\}$. For all $i \in \mathbb{N}$, we let $H_i = A \cup B_i \cup C_i \cup D_i \cup E_i \cup F_i$. We first prove that $S_{i+1}' \subseteq H_{i+1}$. For this, we remark that from the induction hypothesis, $S_i' = H_i \subseteq H_{i+1}$ and by definition $S_{i+1}' = S_i' \cup \{u | S_i' \Rightarrow_R u, R \in \mathsf{INTRUDER}\}$. Thus, it suffices to check that $\{u | S_i' \Rightarrow_R u, R \in \mathsf{INTRUDER}\} \subseteq H_{i+1}$ by reviewing all the rules of $\mathsf{INTRUDER}$ in order to prove that $S_{i+1}' \subseteq H_{i+1}$:

- pair creation : this corresponds to the set B_{i+1}.
- pair destruction : as $S_i' = H_i$, every pair of S_i' is either given by the set B_i or is in S_∞. In all cases, an application of the pair destruction rule is going to create an element of $S_{i-1}' \subseteq S_i' \subseteq H_{i+1}$.

- symmetric encryption : this corresponds to the set C_{i+1}.
- symmetric decryption : because of (5) and $S \not\vdash^* k'_1$, k'_1 (with the notation of the statement of the lemma) is not in S'_i. By applying the induction hypothesis for $S'_i = H_i$, the decryption rule can only apply to an encrypted term of C_i or S_∞. For the first case, this gives elements of $S'_{i-1} \subseteq S'_i \subseteq H_{i+1}$. In the second case, the decryption rule is applied to a term of the form $\{\!|m|\!\}_k$. Assertion (5) and the fact that the elements of NK do not appear in S yield $k \in S_\infty$. Thus the definition of S_∞ implies $m \in S_\infty \subseteq H_{i+1}$.
- asymmetric encryption : this corresponds to the set D_{i+1}.
- asymmetric decryption : because of (5) and $S \not\vdash^* k'_1$, k'_1 is not in S'_i. Thus, by applying the induction hypothesis $S'_i \subseteq H_i$, the decryption rule can only apply to an encrypted term of the set D_i or to a term of S_∞. In the first case, we obtain elements of $S'_{i-1} \subseteq S'_i \subseteq H_{i+1}$. In the second case, because of the fact that the elements of NK do not appear in S and assertion (5), the term is necessarily of the form $\{m\}_k$ with $inv(k) \in S_\infty$. Thus, the definition of S_∞ provides $m \in S_\infty \subseteq H_{i+1}$.
- signature: this corresponds to the set E_{i+1}.
- signature decomposition: by application of the induction hypothesis $S'_i \subseteq H_i$, the signature decomposition rule can only apply to a signed term of S_∞, to the set E_i or to $\Sigma(t_2, k_2)$. In the first case, this gives a term of S_∞, in the second case, we obtain a term of $S'_{i-1} \subseteq S'_i \subseteq H_{i+1}$ and in the last case, this gives t_2 which is by hypothesis deducible from $S_\infty \cup \{f(t_1, k_1)\} \cup \mathsf{NK}$. As this last set is included in A, $t_2 \in H_{i+1}$.
- message encapsulation: this corresponds to the set F_{i+1}.
- message decapsulation: as $S'_i = H_i$, every encapsulated term of S'_i is either given by the set F_i or is in S_∞. In all cases, an application of the decapsulation rule creates an element of $S'_{i-1} \subseteq S'_i \subseteq H_{i+1}$.

We can now prove the other assertions:

- assertion (5) : as $S'_{i+1} \subseteq H'_{i+1}$, we know that S'_{i+1} is included in the union of $A = S_\infty \cup f(t_1, k_1) \cup \Sigma(t_2, k_2) \cup \mathsf{Hdls} \cup \mathsf{NK} \subseteq S'_i$ and of non-atomic terms. Since $A \subseteq S'_i$, we conclude by applying the induction hypothesis (5).
- assertion (7) : if $\{\!|v|\!\}_w \in S'_{i+1}$ and $w \notin S'_{i+1}$ then $\{\!|v|\!\}_w$ is not in the set C_{i+1} and thus $\{\!|v|\!\}_w$ is in the set A which allows to conclude.
- assertion (8) : if $\{v\}_w \in S'_{i+1}$ and $v \notin S'_{i+1}$ then $\{v\}_w$ is not in the set D_{i+1} and thus $\{v\}_w$ is in the set A which allows to conclude.
- assertion (6) : ditto the preceding proof.

In order to finish the proof, we just have to remark that by definition $\cup_{i\in\mathbb{N}} S'_i = \{u | S' \vdash^* u\}$. □

First, we use Lemma 1 in order to check that every initial state (see Definition 1) satisfies the properties (**Sec***), (**Cert**), (**SymEnc**), (**AsymEnc**) (**Sign**). Let $\mathcal{S}_0$ be an initial state. Let

$$A = \{m, N_1, N_2 | h^\alpha_a(N_1, N_2, m, ., ., S_1, S_2) \in \mathcal{S}_0, S_1 \subsetneq H \text{ or } S_2 \subsetneq H, a \in \mathsf{Agent}\} \cup \{m, N_1, N_2 | h^\alpha_a(N_1, N_2, m, ., ., ., .) \in \mathcal{S}_0, a \notin H\} \cup \{m | h_a(., m) \in \mathcal{S}_0\}.$$

Now writing $\mathcal{S}_{0,b}$ for b's knowledge in $\mathcal{S}_0$, we let

$$\mathsf{NK} = A \cup_{b \in \mathsf{Agent} \cup \{\mathsf{int}\}} (\mathcal{S}_{0,b} - \mathsf{Handle}),$$

$$\mathsf{Hdls} = \cup_{b \in \mathsf{Agent} \cup \{\mathsf{int}\}} \mathcal{S}_{0,b} \cap \mathsf{Handle}.$$

Let $S' = \mathsf{NK} \cup \mathsf{Hdls}$. It is clear from the definition of $\vdash^*$ that $\{u | \mathcal{S}_0 \vdash^* u\} = \{u | S' \vdash^* u\}$. Let $h \in \mathsf{Handle}$ be so that $S' \vdash^* h$, from Lemma 1 conclusion (1) (since $S = \emptyset$, we have $h \in \mathsf{Hdls}$. Suppose now that $h = h_a^\alpha(N_1, N_2, m, ., ., S_1, S_2)$. Lemma 1 conclusion (1) tells us that if $S' \vdash^* m$ then $m \in \mathsf{NK}$ which is only possible if $\{a\} \cup S_1 \cup S_1 \not\subseteq H$ from the Definition 1. We deduce that for all $h_a^\alpha(N_1, N_2, m, ., ., S_1, S_2) \in \mathsf{Handle}$ so that $\{a\} \cup S_1 \cup S_2 \subseteq H$, $S' \not\vdash^* m$ and $m \in \mathsf{Keyv} \cup \mathsf{Noncev}$ which is exactly property (**Sec***). By applying Lemma 1 conclusion (2), we obtain that if $S' \vdash^* \{\!|v|\!\}_w$ then necessarily $S' \vdash^* w$ which proves (**SymEnc**). By applying Lemma 1 conclusion (3), we obtain that if $S' \vdash^* \{v\}_w$ then necessarily $S' \vdash^* v$ which proves (**AsymEnc**). Ditto for the signature, Lemma 1 conclusion (4) tells us that if $S' \vdash^* \Sigma(u, k)$ then $S' \vdash^* k$ which proves property (**Sign**). To finish, (**Cert**) is an immediate consequence of the definition of an initial state and of the fact that every deducible handle of S' is in Hdls from Lemma 1 conclusion (1).

We are ready to prove Theorem 1. Let $\mathcal{S}$ be a state satisfying the properties (**Sec***), (**Cert**), (**SymEnc**), (**AsymEnc**), (**Sign**), and let $\mathcal{S}'$ be such that $\mathcal{S} \Rightarrow_R \mathcal{S}'$ for $R \in \mathsf{API} \cup \mathsf{INTRUDER} \cup \mathsf{CONTROL}$. We want to prove that $\mathcal{S}'$ also satisfies the properties.

It is immediate to see that the properties (**Sec***), (**Cert**), (**SymEnc**), (**AsymEnc**), (**Sign**) are invariant by application of the rules of $\mathsf{INTRUDER}$ and $\mathsf{CONTROL}$. Indeed, these properties are about terms deducible from $\mathcal{S}$ but by definition of the deducibility relation, we have $\{u | \mathcal{S} \vdash^* u\} = \{u | \mathcal{S}' \vdash^* u\}$ with $\mathcal{S} \Rightarrow_R \mathcal{S}'$ and $R \in \mathsf{INTRUDER}$. Ditto from the definition of the extended deducibility relation, we have $\{u | \mathcal{S} \vdash^* u\} = \{u | \mathcal{S}' \vdash^* u\}$ with $\mathcal{S} \Rightarrow_R \mathcal{S}'$ and $R \in \mathsf{CONTROL}$.

It only remains to see that the properties are left invariant if $R \in \mathsf{API}$ which is done by checking each rule. In the following, we suppose that $\mathcal{S}$ satisfies the properties and we define $\mathcal{S}'$ as $\mathcal{S} \Rightarrow_R \mathcal{S}'$ with $R \in \mathsf{API}$.

The rule (Sym Gen) : Reminder of the rule (with the notations of the Section 3.2) :

$$i, S_1, S_2 \quad \stackrel{N, X_k}{\Longrightarrow} \quad P_e(h_e^g(N, \mathsf{Null}, X_k, \mathrm{symEncDec}, i, S_1, S_2 \quad \cup \quad \{e\}))$$

We apply Lemma 1 to $\mathcal{S}'_{\mathsf{int}} = \mathcal{S}_{\mathsf{int}} \cup \{h_e^g(N, \mathsf{Null}, X_k, \mathrm{symEncDec}, i, S_1, S_2 \cup \{e\})\} \cup \mathsf{NK}$ with $\mathsf{NK} = \{X_k\}$ if $\{e\} \cup S_1 \cup S_2 \not\subseteq H$ and $\mathsf{NK} = \emptyset$ on the contrary.

- invariance of (**Sec***) : let $h_a^\alpha(., ., m, ., i, S'_1, S'_2) \in \mathsf{Handle}$ with $a \in H$, $S'_1, S'_2 \subseteq H$ be such that $\mathcal{S}' \vdash^* h_a^\alpha(., ., m, ., i, S'_1, S'_2)$ then conclusion (1) of Lemma 1 tells us that either
  - $\mathcal{S} \vdash^* h_a^\alpha(., ., m, ., i, S'_1, S'_2)$,
  - or $h_a^\alpha(., ., m, ., i, S'_1, S'_2) = h_e^g(N, \mathsf{Null}, X_k, \mathrm{symEncDec}, i, S_1, S_2 \cup \{e\})$.

  In the first case, the induction hypothesis allows us to conclude. In the second case, Lemma 1 conclusion (1) with $u = X_k$ implies that $\mathcal{S}' \not\vdash^* X_k$ and on the other side it is clear that $X_k \in \mathsf{Keyv}$ and $e \in S_2 \cup \{e\}$.

- invariance of (**Cert**) : from the conclusion (1) of Lemma 1, if

  $$\mathcal{S}' \vdash^* h_a(N_1, \mathcal{C}(N_1, N_2, N_3, X_k, pub\Theta, i, S'_1, S'_2))$$

  then $\mathcal{S} \vdash^* h_a(N_1, \mathcal{C}(N_1, N_2, N_3, X_k, pub\Theta, i, S'_1, S'_2))$ and the induction hypothesis allows us to conclude.

- invariance of (**SymEnc**) : from the conclusion (2) of Lemma 1, if $\mathcal{S}' \vdash^* \{\!|u|\!\}_k$ and $\mathcal{S}' \nvdash^* k$ then $\mathcal{S} \vdash^* \{\!|u|\!\}_k$ and the induction hypothesis concludes.
- invariance of (**AsymEnc**) : from the conclusion (3) of Lemma 1, if $\mathcal{S}' \vdash^* \{u\}_k$ and $\mathcal{S}' \nvdash^* u$ then $\mathcal{S} \vdash^* \{u\}_k$ and the induction hypothesis concludes.
- invariance of (**Sign**) : from conclusion (4) of Lemma 1, $\mathcal{S}' \vdash^* \Sigma(u, inv(k))$ and $\mathcal{S}' \nvdash^* inv(k)$ implies that $\mathcal{S} \vdash^* \{\!|u|\!\}_k$ and the induction hypothesis conclude.

The rule (Sym Encrypt) : Reminder of the rule (with the notations of the Section 3.2) :

$$P_b(h_b^\alpha(N, \mathsf{Null}, X_k, \text{symEncDec}, i, S_1, S_2)), P_b(m_1), \ldots, P_b(m_n)$$
$$\Longrightarrow P_b(\{\!|m'_1, \ldots, m'_n|\!\}_{X_k})$$

If $\mathcal{S} \vdash^* X_k$, then from hypothesis (**Sec***), it means that either $b \notin H$, or $S_1, S_2 \nsubseteq H$. In this case, if m_j is a handle of the form $h_b^\alpha(N_j, N'_j, X_{k,j}, T_j, i_j, S_{j,1}, S_{j,2})$ with $(i_j, S_{j,1}, S_{j,2}) \prec (i, S_1, S_2)$, it is easy to see that $\mathcal{S} \vdash^* m'_j$ since $\mathcal{S} \vdash^* N_j, X_{k,j}, N'_j$. If m_j is not a handle, we also have $\mathcal{S} \vdash^* m'_j$. From all this, we deduce that $\mathcal{S} \vdash^* \{\!|m'_1, \ldots, m'_n|\!\}_{X_k}$ and the rule does not change the set of deducible terms. This proves the invariance of the induction hypothesis in this case.

We suppose in the following that $\mathcal{S} \nvdash^* X_k$ which allows us to apply Lemma 1 to $\mathcal{S}'_{\mathsf{int}} = \mathcal{S}_{\mathsf{int}} \cup \{\!|m'_1, \ldots, m'_n|\!\}_{X_k}\}$.

- invariance of (**Sec***) : from Lemma 1 conclusion (1), if $h_a^\alpha(., ., m, ., i, S'_1, S'_2) \in \mathsf{Handle}$ is such that $a \in H$, $S'_1, S'_2 \subseteq H$ and $\mathcal{S}' \vdash^* h_a^\alpha(., ., m, ., i, S'_1, S'_2)$, then $\mathcal{S} \vdash^* h_a^\alpha(., ., m, ., i, S'_1, S'_2)$ and the induction hypothesis allows us to conclude.
- invariance of (**Cert**) : ditto, from Lemma 1 conclusion (1), if

  $$\mathcal{S}' \vdash^* h_a(N_1, \mathcal{C}(N_1, N_2, N_3, X_k, pub\Theta, i, S'_1, S'_2))$$

  with $a \in H$, $S'_1, S'_2 \subseteq H$ then $\mathcal{S} \vdash^* h_a(N_1, \mathcal{C}(N_1, N_2, N_3, X_k, pub\Theta, i, S'_1, S'_2))$ and we conclude by using the induction hypothesis.
- invariance of (**SymEnc**) : From Lemma 1 conclusion (2) if $\mathcal{S}' \vdash^* \{\!|u|\!\}_k$ then either
  - $\mathcal{S} \vdash^* \{\!|u|\!\}_k$ and we conclude by using the induction hypothesis ;
  - or we have $\{\!|u|\!\}_k = \{\!|m'_1, \ldots, m'_n|\!\}_{X_k}$, which is built following the condition of the hypothesis.
- invariance of (**AsymEnc**) : from the conclusion (3) of Lemma 1, if $\mathcal{S}' \vdash^* \{u\}_k$ and $\mathcal{S}' \nvdash^* u$ then $\mathcal{S} \vdash^* \{u\}_k$ and the induction hypothesis concludes.
- invariance of (**Sign**) : from Lemma 1 conclusion (4), $\mathcal{S}' \vdash^* \Sigma(u, k)$ and $\mathcal{S}' \nvdash^* k$ implies that $\mathcal{S} \vdash^* \Sigma(u, k)$ and the induction hypothesis allows us to conclude.

The rule (Sym Decrypt) : Reminder of the rule (with the notations of the Section 3.2) :

$$P_b(h_b^\alpha(N, \mathsf{Null}, X_k, \text{symEncDec}, i, S_1, S_2)), P_b(\{\!|m_1, \ldots, m_n|\!\}_{X_k})$$
$$\Longrightarrow P_b(m'_1), \ldots, P_b(m'_n)$$

We remark that if $\mathcal{S} \vdash^* X_k$, then the rule creates handles and under which are stored terms which are already deducible from $\mathcal{S}$.

If $\mathcal{S} \nvdash^* X_k$, the by applying the hypothesis (**SymEnc**) to $\mathcal{S} \vdash^* \{\!|m_1, \ldots, m_n|\!\}_{X_k}$, we obtain that if m'_j is not a handle then $\mathcal{S} \vdash^* m'_j$. In all cases, the rule builds either terms which are deducible from $\mathcal{S}$,

or handles. We can thus consider that the rule only creates handles. Let $h_b^\alpha(.,.,m,.,i_c,S_1',S_2') \in \mathsf{Handle}$ be such a handle. In order to be able to apply Lemma 1, we want to show that if

$$\{b\} \cup S_1' \cup S_2' \not\subseteq H \tag{15}$$

then $\mathcal{S} \vdash^* m$. Indeed,

- if $\mathcal{S} \vdash^* X_k$ then $\mathcal{S} \vdash^* m$;
- if $\mathcal{S} \nvdash^* X_k$, then from property (**SymEnc**), there exists $c \in \mathsf{Agent}$ such that

  $$\mathcal{S} \vdash^* h_c^\alpha(.,.,m,.,i_c,S_1',S_2').$$

  we know that $b \in H$ (the contrary would contradict $\mathcal{S} \nvdash^* X_k$). Furthermore, we have $S_1' \cup S_2' \not\subseteq H$ since we have $\{b\} \cup S_1' \cup S_2' \not\subseteq H$ and hence we deduce that $\mathcal{S} \vdash^* m$ which is what we want.

This allows us to use Lemma 1 with $\mathcal{S}'_{\mathsf{int}} = \mathcal{S}_{\mathsf{int}} \cup \mathsf{Hdls}$ where Hdls are the new handles created by the rule.

- invariance of (**Sec***) : let $h_a^\alpha(.,.,m,.,i,S_1',S_2') \in \mathsf{Handle}$ with $a \in H$, $S_1', S_2' \subseteq H$, from Lemma 1 conclusion (1), we have $\mathcal{S}' \vdash^* h_a^\alpha(.,.,m,.,i,S_1',S_2')$ with $a \in H$, $S_1', S_2' \subseteq H$, implies that either
  - $\mathcal{S} \vdash^* h_a^\alpha(.,.,m,.,i,S_1',S_2')$ and we conclude by appealing to the induction hypothesis ;
  - or $h_a^\alpha(.,.,m,.,i,S_1',S_2')$ is m_j' a handle created by applying the rule (**Sym Decrypt**) on $\{\!|m_1,\ldots,m_n|\!\}_{X_k}$.

  In this last case, the handle m_j' is of the form $h_b^r(N_j,N_j',X_{k,j},T_j,i_j,S_{j,1},S_{j,2})$. We have to show that if $b \in H$ and $S_{j,1}, S_{j,2} \subseteq H$ then $\mathcal{S}' \nvdash^* X_{k,j}$. We thus suppose that $b \in H$ and $S_{j,1}, S_{j,2} \subseteq H$. As $(i_j, S_{j,1}, S_{j,2}) \prec (i, S_1, S_2)$, by applying the hypothesis (**Sec***), we deduce that $\mathcal{S} \nvdash^* X_k$. Thus we can apply hypothesis (**SymEnc**) to $\mathcal{S} \vdash^* \{\!|m_1,\ldots,m_n|\!\}_{X_k}$. We obtain the existence of a $c \in H$ such that

  $$\mathcal{S} \vdash^* h_c^\alpha(N, \mathsf{Null}, X_k, \mathrm{symEncDec}, i_c, S_{c,1}, S_{c,2})$$

  and furthermore the existence of a handle $h_c(.,.,X_{k,j},.,i_j,S_{j,1},S_{j,2})$ with $X_{k,j} \in \mathsf{Keyv} \cup \mathsf{Noncev}$. We know that $\{c\} \cup S_{j,1} \cup S_{j,2} \subseteq H$ and by applying (**Sec***), we obtain that $\mathcal{S} \nvdash^* X_{k,j}$. Now, from conclusion (1) of Lemma 1, we deduce that $\mathcal{S}' \nvdash^* X_{k,j}$ and $X_{k,j} \in \mathsf{Keyv} \cup \mathsf{Noncev}$ and that $b \in S_{j,2}$.

For the other hypothesis, properties (**Cert**), (**SymEnc**), (**AsymEnc**), (**Sign**) for $\mathcal{S}'$ reduces, by immediate application of respectively the conclusion (1), (2), (3) (4) of Lemma 1, to the same properties for $\mathcal{S}$.

The rule (Asym Gen) : Reminder of the rule (with the notations of Section 3.2) :

$$\begin{array}{l} P_e(h_e^\alpha(N_1,N_2,inv(Y_k),\mathrm{privCertSign},i_1,S_{e,1},S_{e,2})),\\ \quad P_e(h_e(N_2,\mathcal{C}(N_2,N_1,N_{cert},Y_k,\mathrm{pubCertVerif},i_1,S_{e,1},S_{e,2}))),i_2,S_1,S_2,b \overset{N_3,N_4,X_k}{\Longrightarrow}\\ \quad P_e(h_e^g(N_3,N_4,inv(X_k),\mathrm{privDecSign},i_2,S_{e,1}\cup S_{e,2}\cup S_1\cup\{e\},\{b,e\}\cup S_2)),\\ P_e(\Sigma(\mathcal{C}(N_4,N_3,N_2,X_k,\mathrm{pubEncVerif},i_2,S_{e,1}\cup S_{e,2}\cup S_1\cup\{e\},\{b,e\}\cup S_2),inv(Y_k))) \end{array}$$

Let $S'_{e,1} = S_{e,1} \cup S_{e,2} \cup S_1 \cup \{e\}$ and $S'_{e,2} = \{b,e\} \cup S_2$ be the control sets of the handle created by the rule. In the proof of the preservation of the invariants, we use two instances of Lemma 1, according to whether $\mathcal{S} \vdash^* inv(Y_k)$ or $\mathcal{S} \nvdash^* inv(Y_k)$:

- In case $\mathcal{S} \vdash^* inv(Y_k)$, it follows from the induction hypothesis (**Sec***) that $\{e\} \cup S_{e,1} \cup S_{e,2} \not\subseteq H$, and thus $\{e\} \cup S'_{e,1} \cup S'_{e,2} \not\subseteq H$. Then, we apply Lemma 1 to

  $$\mathcal{S}''_{\mathsf{int}} = \mathcal{S}_{\mathsf{int}} \cup \{h^g_e(N_3, N_4, inv(X_k), \text{privDecSign}, i_2, S_{e,1} \cup S_{e,2} \cup S_1 \cup \{e\}, \{b, e\} \cup S_2)\} \cup \mathsf{NK},$$

  where $\mathsf{NK} = \{N_3, N_4, X_k, inv(X_k)\}$. Indeed, it is clear that $N_3, N_4, X_k, inv(X_k)$ do not appear in $\mathcal{S}$ and furthermore $\{u | \mathcal{S}' \vdash^* u\} = \{u | \mathcal{S}''_{\mathsf{int}} \vdash^* u\}$.

- In the case that $\mathcal{S} \nvdash^* inv(Y_k)$, we apply Lemma 1 to

  $$\mathcal{S}'_{\mathsf{int}} = \mathcal{S}_{\mathsf{int}} \cup \{h^g_e(N_3, N_4, inv(X_k), \text{privDecSign}, i_2, S_{e,1} \cup S_{e,2} \cup S_1 \cup \{e\}, \{b, e\} \cup S_2)\} \cup$$
  $$\{\Sigma(\mathcal{C}(N_4, N_3, N_2, X_k, \text{pubEncVerif}, i_2, S_{e,1} \cup S_{e,2} \cup S_1 \cup \{e\}, \{b, e\} \cup S_2), inv(Y_k))\} \cup \mathsf{NK},$$

  with $\mathsf{NK} = \{N_3, N_4, X_k\}$ if $\{e\} \cup S'_{e,1} \cup S'_{e,2} \subseteq H$ and else $\mathsf{NK} = \{N_3, N_4, X_k, inv(X_k)\}$.

- invariance of (**Sec***) : from Lemma 1 conclusion (1), we have that $\mathcal{S}' \vdash^* h^\alpha_a(., ., m, ., i, S'_1, S'_2)$ with $a \in H$, $S'_1, S'_2 \subseteq H$ implies:
  - either $\mathcal{S} \vdash^* h^\alpha_a(., ., m, ., i, S'_1, S'_2)$;
  - or $h^\alpha_a(., ., m, ., i, S'_1, S'_2)$ comes from the rule (**Asym Gen**).

  In the first case, we appeal to the induction hypothesis on $\mathcal{S}$. The second case can only arise when $\mathcal{S} \nvdash^* inv(Y_k)$: indeed we have then $\{e\} \cup S_{e,1} \cup S_{e,2} = \{a\} \cup S'_1 \cup S'_2 \subseteq H$ and our induction hypothesis (**Sec***) yields $\mathcal{S} \nvdash^* inv(Y_k)$. As $inv(X_k) \notin \mathsf{Handle} \cup \{N_3, N_4, X_k\}$, Lemma 1 conclusion (1) gives $\mathcal{S}' \nvdash^* inv(X_k)$. Furthermore, it is clear that $inv(X_k) \in \mathsf{Keyv}$.

- invariance of (**Cert**), (**SymEnc**) and of (**AsymEnc**): none of the terms concerned by these properties are created by this rule.

- invariance of (**Sign**) : if $\mathcal{S} \vdash^* inv(Y_k)$, thanks to Lemma 1 conclusion (4), we easily obtain that a signature deducible from $\mathcal{S}'$ is already deducible from $\mathcal{S}$. If $\mathcal{S} \nvdash^* inv(Y_k)$, from Lemma 1 conclusion (4), we have $\mathcal{S}' \vdash^* \Sigma(t, inv(Y_k))$ implies either
  - $\mathcal{S} \vdash^* \Sigma(t, inv(Y_k))$;
  - or

    $$\Sigma(t, inv(Y_k)) = \Sigma(\mathcal{C}(N_4, N_3, N_2, X_k, \text{pubEncVerif}, i_2, S_{e,1} \cup S_{e,2} \cup S_1, \{b, e\} \cup S_2), inv(Y_k)).$$

  In the first case, the induction hypothesis allows us to conclude and in the second case, it is easily seen that the property (**Sign**) is verified.

The rule (Asym SignEncrypt) : Reminder of the rule (with the notations of Section 3.2) :

$$P_b(h^\alpha_b(N_1, N_2, inv(Y_k), \text{privDecSign}, i_b, S_{b,1}, S_{b,2}),$$
$$P_b(h_b(N_3, \mathcal{C}(N_3, N_4, N_5, X_k, \text{pubEncVerif}, i_c, S_{c,1}, S_{c,2}))),$$
$$P_b(m_1), \ldots, P_b(m_n) \Longrightarrow P_b(\{m'_1, \ldots, m'_n\}_{X_k}), P_b(\Sigma(\{m'_1, \ldots, m'_n\}_{X_k}, inv(Y_k)))$$

Suppose that $\mathcal{S} \vdash^* inv(Y_k)$. From the hypothesis (**Sec***), it means that $S_{b,1} \cup S_{b,2} \cup \{b\} \not\subseteq H$. Suppose now that for $j \in [1..n]$, we have $m_j = h^\alpha_b(N_j, N'_j, X_{k,j}, T_j, i_j, S_{j,1}, S_{j,2})$ and $m'_j \neq \varnothing$. Then by construction, we have

$$(i_j, S_{j,1}, S_{j,2}) \prec (i_b, S_{b,1}, S_{b,2}). \tag{16}$$

Thus, from the definition of $\vdash^*$ and because of (16), we have $\mathcal{S} \vdash^* N_j, X_{k,j}, N'_j$ and thus $\mathcal{S} \vdash^* m'_j$. Since furthermore, $\mathcal{S} \vdash^* m'_j$ if m_j is not a handle, we deduce that $\mathcal{S} \vdash^* \{m'_1, \ldots, m'_n\}_{X_k}$. We have moreover $\mathcal{S} \vdash^* \Sigma(\{m'_1, \ldots, m'_n\}_{X_k}, inv(Y_k))$. In this case the execution of the rule does not change the set of deducible terms.

From now on, we suppose that $\mathcal{S} \nvdash^* inv(Y_k)$. We have two cases:

- If $S_{c,1} \cup S_{c,2} \subseteq H$, by applying the hypothesis (**Cert**), there exists a $c \in S_{c,2}$ and a handle such that:
$$\mathcal{S} \vdash^* h_c^\alpha(N_4, N_3, inv(X_k), \text{privDecSign}, i_c, S_{c,1}, S_{c,2}).$$
By applying (**Sec***), we find that $\mathcal{S} \nvdash^* inv(X_k)$. As moreover $\mathcal{S} \nvdash^* inv(Y_k)$, we can use Lemma 1 with $\mathcal{S}'_{\mathsf{int}} = \mathcal{S}_{\mathsf{int}} \cup \{\{m'_1, \ldots, m'_n\}_{X_k}\} \cup \{\Sigma(\{m'_1, \ldots, m'_n\}_{X_k}, inv(Y_k))\}$.

- If $S_{c,1} \cup S_{c,2} \nsubseteq H$, as by construction we have
$$(i_j, S_{j,1}, S_{j,2}) \prec (i_c, S_{c,1}, S_{c,2}), \tag{17}$$
we deduce as before that $\mathcal{S} \vdash^* N_j, X_{k,j}, N'_j$ and thus $\mathcal{S} \vdash^* m'_j$ if m_j is a handle. As moreover, $\mathcal{S} \vdash^* m'_j$ if m_j is not a handle, we deduce that $\mathcal{S} \vdash^* \{m'_1, \ldots, m'_n\}_{X_k}$. We can then use Lemma 1 with $\mathcal{S}'_{\mathsf{int}} = \mathcal{S}_{\mathsf{int}} \cup \{\Sigma(\{m'_1, \ldots, m'_n\}_{X_k}, inv(Y_k))\}$.

- invariance of (**Sec***) : from Lemma 1 conclusion (1), if there exists a handle $h_a^\alpha(., ., m, ., i, S'_1, S'_2)$ such that
$$\mathcal{S}' \vdash^* h_a^\alpha(., ., m, ., i, S'_1, S'_2)$$
then $\mathcal{S} \vdash^* h_a^\alpha(., ., m, ., i, S'_1, S'_2)$ and the induction hypothesis allows us to conclude.

- invariance of (**Cert**) : ditto, Lemma 1 conclusion (1) allows us to reduce directly to the induction hypothesis.

- invariance of (**SymEnc**) : Lemma 1 conclusion (2) allows us to reduce to the induction hypothesis.

- invariance of (**AsymEnc**) : Let $u, k \in \mathsf{Msg}$ be such that $\mathcal{S}' \vdash^* \{u\}_k$ and $\mathcal{S}' \nvdash^* u$ then Lemma 1 conclusion (3) tells us that either
  - $\mathcal{S} \vdash^* \{u\}_k$,
  - or $\{u\}_k = \{m'_1, \ldots, m'_n\}_{X_k}$.

  In the first case we are reduced to the induction hypothesis, and in the second case (**AsymEnc**) is easily verified.

- invariance of (**Sign**) : by applying Lemma 1 conclusion (4), we obtain that if $\Sigma(u, k)$ is a term such that $\mathcal{S}' \vdash^* \Sigma(u, k)$ then either $\mathcal{S} \vdash^* \Sigma(u, k)$ and the induction hypothesis allows us to conclude or $\Sigma(u, k)$ is generated by the rule and it is an immediate verification that the hypothesis (**Sign**) is fulfilled in this case.

The rule (Asym VerifDecrypt) : Reminder of the rule (with the notations of the Section 3.2) :

$$\begin{aligned} &P_b(h_b(N_1, \mathcal{C}(N_1, N_2, N_3, Y_k, \text{pubEncVerif}, i_c, S_{c,1}, S_{c,2}))),\\ &\qquad P_b(h_b^\alpha(N_4, N_5, inv(X_k), \text{privDecSign}, i_b, S_{b,1}, S_{b,2})),\\ &\qquad P_b(\{m_1, \ldots, m_n\}_{X_k}), P_b(\Sigma(\{m_1, \ldots, m_n\}_{X_k}, inv(Y_k)))\\ \Longrightarrow &P_b(m'_1), \ldots, P_b(m'_n) \end{aligned}$$

First, we remark that the terms created by the rule which are not handles are deducible from $\mathcal{S}$. In fact, we have either:

- $\mathcal{S} \vdash^* m_1, \ldots, m_n$ and in this case it is clear that if m'_j is not a handle then $\mathcal{S} \vdash^* m'_j$;

- or $\mathcal{S} \nvdash^* m_1, \ldots, m_n$ and in this case, by applying (**AsymEnc**), we obtain that if m'_j is not a handle, there exists $m \in \mathsf{Msg}$ such that $\mathcal{S} \vdash^* m$ and $m'_j = m$.

We have established that the execution of the rule only produces terms which are deducible from $\mathcal{S}$ and handles. Let $h_b^\alpha(.,.,m,.,i,S_1',S_2')$ be such a handle. To allow us to apply Lemma 1, we are going to prove that if $\{b\} \cup S_1' \cup S_2' \not\subseteq H$ then $\mathcal{S} \vdash^* m$. Indeed, we have either

- $\mathcal{S} \vdash^* m_1, \ldots, m_n$ and it case it is clear that $\mathcal{S} \vdash^* m$;
- or $\mathcal{S} \not\vdash^* m_1, \ldots, m_n$ which implies that $\mathcal{S} \not\vdash^* inv(X_k)$ from which we deduce that $b \in H$. Then using (**AsymEnc**), there exists $c \in \mathsf{Agent}$ and a handle $h_c^\alpha(.,.,m,.,i,S_1',S_2')$ such that $\mathcal{S} \vdash^* h_c^\alpha(.,.,m,.,i,S_1',S_2')$. As $b \in H$ and $\{b\} \cup S_1' \cup S_2' \not\subseteq H$, we deduce that $S_1' \cup S_2' \not\subseteq H$ so that $\mathcal{S} \vdash^* m$.

At this point, we have proven that the execution of the rule only produce terms which are deducible from $\mathcal{S}$ and handles, and that if the value of these handles is deducible from $\mathcal{S}'$ then it was already deducible from $\mathcal{S}$. Let Hdls be the set of handles created by the rule, we can thus apply Lemma 1 with $\mathcal{S}'_{\mathsf{int}} = \mathcal{S}_{\mathsf{int}} \cup \mathsf{Hdls}$.

- invariance of (**Sec***) : let $h_a^\alpha(.,.,m,.,i,S_1',S_2')$ be a handle with $a \in H$, $S_1', S_2' \subseteq H$, from Lemma 1 conclusion (1), we have that $\mathcal{S}' \vdash^* h_a^\alpha(.,.,m,.,i,S_1',S_2')$, implies:
  - either $\mathcal{S} \vdash^* h_a^\alpha(.,.,m,.,i,S_1',S_2')$ and the induction hypothesis allows us to conclude;
  - or $h_a^\alpha(.,.,m,.,i,S_1',S_2')$ is a handle created by application of the rule (**Asym VerifDecrypt**).

  In this latter case, we have

  $$h_a^\alpha(.,.,m,.,i,S_1',S_2') = h_b^r(N_j, N_j', X_{k,j}, T_j, i_j, S_{j,1}, S_{j,2}),$$

  with

  $$(i_j, S_{j,2}, S_{j,2}) \prec (i_b, S_{b,1}, S_{b,2}) \text{ and } (i_j, S_{j,2}, S_{j,2}) \prec (i_c, S_{c,1}, S_{c,2}). \tag{18}$$

  We have to prove that if $b \in H$ and $S_{j,1}, S_{j,2} \subseteq H$ then $\mathcal{S}' \not\vdash^* X_{k,j}$, $X_{k,j} \in \mathsf{Key} \cup \mathsf{Nonce}$ and $b \in S_{j,2}$. Since

  $$\mathcal{S} \vdash^* h_b(N_1, \mathcal{C}(N_1, N_2, N_3, Y_k, \text{pubEncVerif}, i_c, S_{c,1}, S_{c,2})),$$

  from the hypothesis (**Cert**) and (18), we know that $\exists c \in S_{c,2}$ such as

  $$\mathcal{S} \vdash^* h_c(N_2, N_1, inv(Y_k), \text{privDecSign}, i_c, S_{c,1}, S_{c,2}).$$

  From (18), we have $c \in H$ and $S_{c,1}, S_{c,2} \subseteq H$ so that by application of (**Sec***), we obtain that

  $$\mathcal{S} \not\vdash^* inv(Y_k). \tag{19}$$

  Now, thanks to (**Sign**) and (19), we obtain the existence of a handle

  $$\mathcal{S} \vdash^* h_d(N_j, N_j', X_{k,j}, T_j, i_j, S_{j,1}, S_{j,2}),$$

  with $X_{k,j} \in \mathsf{Keyv} \cup \mathsf{Noncev}$. As $d \in H$, $S_{j,1}, S_{j,2} \subseteq H$, by applying (**Sec***), we obtain that $\mathcal{S} \not\vdash^* X_{k,j}$, $X_{k,j} \in \mathsf{Key} \cup \mathsf{Nonce}$ and $b \in S_{j,2}$. From $\mathcal{S} \not\vdash^* X_{k,j}$ and Lemma 1 conclusion (1) applied to $u = X_{k,j}$, we deduce $\mathcal{S}' \not\vdash^* X_{k,j}$.

Preservation of invariants (**SymEnc**), (**AsymEnc**), (**Sign**) and (**Cert**) follow from the induction hypothesis and Lemma 1 applied to $\mathcal{S}' = \mathcal{S} \cup \mathsf{Hdls}$.

The rule (Cert Gen) : Reminder of the rule (with the notations of the Section 3.2) :

$$\begin{array}{l} P_e(h_e^{\alpha}(N_1, N_2, inv(Y_k), \text{privCertSign}, i_e, S_{e,1}, S_{e,2})), \\ \qquad P_e(h_e(N_2, \mathcal{C}(N_2, N_1, N_{cert}, Y_k, \text{pubCertVerif}, i_e, S_{e,1}, S_{e,2}))), i_b, S_1, S_2 \overset{N_3, N_4, X_k}{\Longrightarrow} \\ \qquad P_e(h_e^{g}(N_3, N_4, inv(X_k), \text{privCertSign}, i_b, S_{e,1} \cup S_{e,2} \cup S_1 \cup \{e\}, \{e, b\} \cup S_2)), \\ P_e(\Sigma(\mathcal{C}(N_4, N_3, N_2, X_k, \text{pubCertVerif}, i_b, S_{e,1} \cup S_{e,2} \cup S_1 \cup \{e\}, \{e, b\} \cup S_2), inv(Y_k))) \end{array}$$

The proof for this rule is exactly the same as that of (**Asym Gen**).

The rule (Cert Verif) : Reminder of the rule (with the notations of the Section 3.2) :

$$\begin{array}{l} P_b(\Sigma(\mathcal{C}(N_1, N_2, N_3, X_k, pub\Theta, i_c, S_{c,1}, S_{c,2}), inv(Y_k))), \\ \qquad P_b(h_b(N_3, \mathcal{C}(N_3, N_4, N_5, Y_k, \text{pubCertVerif}, i_e, S_{e,1}, S_{e,2}))) \Longrightarrow \\ P_b(h_b(N_1, \mathcal{C}(N_1, N_2, N_3, X_k, pub\Theta, i_c, S_{c,1}, S_{c,2}))) \end{array}$$

The rule creates a handle the value of which is clearly deducible from $\mathcal{S}$ (by applying a signature deduction rule to the signed term). We can thus apply Lemma 1, by setting $\mathcal{S}'_{\mathsf{int}} = \mathcal{S}_{\mathsf{int}} \cup \{h_b(N_1, \mathcal{C}(N_1, N_2, N_3, X_k, pub\Theta, i_c, S_{c,1}, S_{c,2}))\}$.

- invariance of (**Sec***) : it reduces immediately to the induction hypothesis thanks to Lemma 1 conclusion (1).
- invariance of (**Cert**) : because of Lemma 1 conclusion (1) and the induction hypothesis, we need only prove (**Cert**) for an integrity handle produced by the execution of the rule. In other words, we must prove that if $b \in H$, $S_{c,1} \cup S_{c,2} \subseteq H$ then there exists $c \in S_{c,2}$ such that :

  $$\mathcal{S}' \vdash^* h_c^{\cdot}(N_2, N_1, inv(X_k), \mathsf{priv}\Theta, i_c, S_{c,1}, S_{c,2}). \tag{20}$$

  As we consider a handle created by the rule, we have

  $$(i_c, S_{c,1}, \varnothing) \prec (i_e, S_{e,1} \cup S_{e,2}, \varnothing), \tag{21}$$

  Thus, $S_{e,1} \cup S_{e,2} \subseteq H$. By applying the hypothesis (**Cert**) to

  $$\mathcal{S} \vdash^* h_b(N_3, \mathcal{C}(N_3, N_4, N_5, Y_k, S_{e,1}, \text{pubCertVerif}, i_e, S_{e,2})),$$

  we deduce the existence of $e \in S_{e,2}$ such that

  $$\mathcal{S} \vdash^* h_e^{\cdot}(N_4, N_3, inv(Y_k), \text{privCertSign}, i_e, S_{e,1}, S_{e,2}).$$

  Because of (21), we have $e \in H$ and $S_{e,1} \cup S_{e,2} \subseteq H$, so that from (**Sec***), $\mathcal{S} \nvdash^* inv(Y_k)$. By applying (**Sign**) to

  $$\mathcal{S} \vdash^* \Sigma(\mathcal{C}(N_1, N_2, N_3, X_k, pub\Theta, i_c, S_{c,1}, S_{c,2}), inv(Y_k)),$$

  we deduce (20) with $c \in S_{c,2}$.
- invariance of (**SymEnc**) : it directly follows from Lemma 1 conclusion (2).
- invariance of (**AsymEnc**) : it directly follows from Lemma 1 conclusion (3).
- invariance of (**Sign**) : it directly follows from Lemma 1 conclusion (4).

□

6 Experiments

We have used our API to implement some asymmetric key protocols based on well-known examples from the Clark-Jacob corpus. Since we impose a secure encryption and signature scheme, our versions of protocols are secure even when the original is not. For example, our implementation of Needham-Schroeder public key avoids Lowe's attack because all messages are signed. Full details together with a Prolog script for generating API commands from protocols are available at `http://www.lsv.ens-cachan.fr/~steel/genericapi/asym`.

7 Conclusions

We have given the design for a key management API for cryptographic devices that allows the use of asymmetric keys for managing keys, together with security properties and proofs in the Dolev Yao model. This is first such design with security proofs as far as we are aware. A variant of this API will be soon implemented for use in French ministry of defence systems. In future work we will add more flexibility to the API. In particular it should be easy to adapt the design to other security orderings not necessarily based on agent identifiers.

References

[1] Martin Abadi and Phillip Rogaway. Reconciling two views of cryptography. In Jan van Leeuwen, Osamu Watanabe, Masami Hagiya, Peter Mosses, and Takayasu Ito, editors, *Theoretical Computer Science: Exploring New Frontiers of Theoretical Informatics*, volume 1872 of *Lecture Notes in Computer Science*, pages 3–22. Springer Berlin / Heidelberg, 2000.

[2] M. Bond. Attacks on cryptoprocessor transaction sets. In *Proceedings of the 3rd International Workshop on Cryptographic Hardware and Embedded Systems (CHES'01)*, volume 2162 of *LNCS*, pages 220–234, Paris, France, 2001. Springer.

[3] Matteo Bortolozzo, Matteo Centenaro, Riccardo Focardi, and Graham Steel. Attacking and fixing PKCS#11 security tokens. In *Proceedings of the 17th ACM Conference on Computer and Communications Security (CCS'10)*, pages 260–269, Chicago, Illinois, USA, October 2010. ACM Press.

[4] C. Cachin and N. Chandran. A secure cryptographic token interface. In *Computer Security Foundations (CSF-22)*, pages 141–153, Long Island, New York, 2009. IEEE Computer Society Press.

[5] J. Clulow. The design and analysis of cryptographic APIs for security devices. Master's thesis, University of Natal, Durban, 2003.

[6] J. Clulow. On the security of PKCS#11. In *Proceedings of CHES 2003*, pages 411–425, 2003.

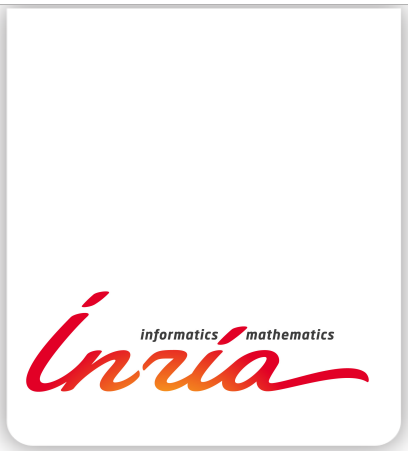
[7] Véronique Cortier and Graham Steel. A generic security api for symmetric key management on cryptographic devices. In Michael Backes and Peng Ning, editors, *Computer Security - ESORICS 2009*, volume 5789 of *Lecture Notes in Computer Science*, pages 605–620. Springer Berlin / Heidelberg, 2009.

[8] Véronique Cortier, Graham Steel, and Cyrille Wiedling. Revoke and let live: A secure key revocation API for cryptographic devices. In *19th ACM Conference on Computer and Communications Security (CCS'12)*, Raleigh, USA, October 2012. ACM.

[9] Danny Dolev and Andrew Chi-Chih Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–207, 1983.

[10] Steve Kremer, Robert Knnemann, and Graham Steel. Universally composable key-management. *IACR Cryptology ePrint Archive*, 2012:189, 2012.

[11] Steve Kremer, Graham Steel, and Bogdan Warinschi. Security for key management interfaces. In *Proceedings of the 24th IEEE Computer Security Foundations Symposium (CSF'11)*, pages 266–280, Cernay-la-Ville, France, June 2011. IEEE Computer Society Press.



RESEARCH CENTRE
PARIS – ROCQUENCOURT

Domaine de Voluceau, - Rocquencourt
B.P. 105 - 78153 Le Chesnay Cedex

Publisher
Inria
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
inria.fr

ISSN 0249-6399